

PCT

特許協力条約に基づいて公開された国際出願



<p>(51) 国際特許分類6 <b>G06F 9/45</b></p>	<p><b>A1</b></p>	<p>(11) 国際公開番号 <b>WO99/30231</b></p> <p>(43) 国際公開日 1999年6月17日(17.06.99)</p>
<p>(21) 国際出願番号 PCT/JP97/04542</p> <p>(22) 国際出願日 1997年12月10日(10.12.97)</p> <p>(71) 出願人 (米国を除くすべての指定国について) 株式会社 日立製作所(HITACHI, LTD.)(JP/JP) 〒101 東京都千代田区神田駿河台四丁目6番地 Tokyo, (JP)</p> <p>(72) 発明者; および (75) 発明者/出願人 (米国についてのみ) 本川敏子(MOTOKAWA, Keiko)(JP/JP) 〒216 神奈川県川崎市宮前区宮前平1丁目7番2号 Kanagawa, (JP) 西山博泰(NISHIYAMA, Hiroyasu)(JP/JP) 〒216 神奈川県川崎市宮前区神木本町1丁目2番3号 プラザ神木B-205 Kanagawa, (JP) 菊池純男(KIKUCHI, Sumio)(JP/JP) 〒195 東京都町田市三輪緑山3丁目22番8号 Tokyo, (JP)</p> <p>(74) 代理人 弁理士 小川勝男(OGAWA, Katsuo) 〒100 東京都千代田区丸の内一丁目5番1号 株式会社 日立製作所内 Tokyo, (JP)</p>		<p>(81) 指定国 JP, US, 欧州特許 (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).</p> <p>添付公開書類 国際調査報告書</p>
<p>(54)Title: <b>MEMORY ACCESS OPTIMIZING METHOD</b></p> <p>(54)発明の名称 メモリアクセス最適化方法</p> <p>(57) Abstract</p> <p>A memory access optimizing method which judges an access method suitable for each memory access and executes the preload optimization and prefetch optimization, according to the judgement result, using an architecture equipped with a prefetch mechanism to write the data on a main storage into a cache and a preload mechanism to write the data on the main storage into a register without writing it into the cache. The memory access method judging step analyzes whether there is a designation of memory access method by a user or not. Moreover, the memory access method judging step investigates whether the data are already in cache or not, whether the data compete with other data for cache or not, whether the data are to be referred to again later or not, and whether the data meet the restriction on register resource or not.</p> <div style="display: flex; align-items: center;"> <div style="flex: 1;"> <pre> graph TD     Start([start 開始]) --&gt; 201[構文解析]     201 --&gt; 202[メモリアクセス方法判定]     202 --&gt; 203[プリロード最適化]     203 --&gt; 204[プリフェッチ最適化]     204 --&gt; 205[コード生成]     205 --&gt; End([end 終了])                     </pre> </div> <div style="flex: 1; padding-left: 20px;"> <p>201 ... analysis of sentence construction</p> <p>202 ... judgment of memory access method</p> <p>203 ... preload optimization</p> <p>204 ... prefetch optimization</p> <p>205 ... code generation</p> </div> </div>		

(57)要約

主記憶上のデータをキャッシュに書き込むプリフェッチ機構と、主記憶上のデータをキャッシュに書き込まずにレジスタに書き込むプリロード機構とを備えたアーキテクチャを対象として、各メモリアクセスに適したアクセス方法を判定し、その判定結果に従ってプリロード最適化およびプリフェッチ最適化を実施するメモリアクセス最適化方法を提供する。メモリアクセス方法判定ステップは、ユーザによるメモリアクセス方法の指定があるかどうかを解析する。また、メモリアクセス方法判定ステップは、データが既にキャッシュにあるかどうか、他のデータとキャッシュ競合するか、データが後で再参照されるか、レジスタ資源の制約を満たすかを調べる。

PCTに基づいて公開される国際出願のパンフレット第一頁に掲載されたPCT加盟国を同定するために使用されるコード(参考情報)

AE アラブ首長国連邦	ES スペイン	LI リヒテンシュタイン	SG シンガポール
AL アルバニア	FI フィンランド	LK スリ・ランカ	SI スロヴェニア
AM アルメニア	FR フランス	LR リベリア	SK スロヴァキア
AT オーストリア	GA ガボン	LS レソト	SL シエラ・レオネ
AU オーストラリア	GB 英国	LT リトアニア	SN セネガル
AZ アゼルバイジャン	GD グレナダ	LU ルクセンブルグ	SZ スワジランド
BA ボスニア・ヘルツェゴビナ	GE グルジア	LV ラトヴィア	TD チャード
BB バルバドス	GH ガーナ	MC モナコ	TG トーゴ
BE ベルギー	GM ガンビア	MD モルドヴァ	TJ タジキスタン
BF ブルキナ・ファソ	GN ギニア	MG マダガスカル	TM トルクメニスタン
BG ブルガリア	GW ギニア・ビサウ	MK マケドニア旧ユーゴスラヴィア	TR トルコ
BJ ベナン	GR ギリシャ	共和国	TT トリニダード・トバゴ
BR ブラジル	HR クロアチア	マリ	UA ウクライナ
BY ベラルーシ	HU ハンガリー	ML モンゴル	UG ウガンダ
CA カナダ	ID インドネシア	MN モンゴリア	US 米国
CF 中央アフリカ	IE アイルランド	MW マラウイ	UZ ウズベキスタン
CG コンゴ	IL イスラエル	MX メキシコ	VN ヴェトナム
CH スイス	IN インド	NE ニジェール	YU ユーゴスラビア
CI コートジボアール	IS アイスランド	NL オランダ	ZA 南アフリカ共和国
CM カメルーン	IT イタリア	NO ノールウェー	ZW ジンバブエ
CN 中国	JP 日本	NZ ニュー・ジージーランド	
CU キューバ	KE ケニア	PL ポーランド	
CY キプロス	KG キルギスタン	PT ポルトガル	
CZ チェッコ	KP 北朝鮮	RO ルーマニア	
DE ドイツ	KR 韓国	RU ロシア	
DK デンマーク	KZ カザフスタン	SD スーダン	
EE エストニア	LC セントルシア	SE スウェーデン	

## 明 細 書

### メモリアクセス最適化方法

#### 技術分野

本発明は、キャッシュ記憶をそなえたプロセッサをターゲットとするコンパイラに関し、特に配列要素参照のロード方式を最適化するメモリアクセス最適化方法に関する。

#### 背景技術

マイクロプロセッサの速度向上に伴って主記憶アクセスのレイテンシが増大し、プログラムの実行性能に与える影響が増している。多くのプロセッサでは、主記憶よりもアクセスが速く比較的小容量のキャッシュ記憶を備え、レイテンシの大きい主記憶へのアクセス回数を削減している。すなわちメモリアクセス命令は、キャッシュヒット時には短いレイテンシでキャッシュをアクセスし、キャッシュミス時にのみ、主記憶をアクセスする。

キャッシュミス時のメモリアクセスのレイテンシを隠す1つの手法として、プリフェッチ（ソフトウェア・プリフェッチ）がある。プリフェッチ命令は、主記憶上のデータをノンブロッキングでキャッシュ上にロードする。コンパイラでは、ソフトウェアパイプライン等のスケジューリング技法を用いて、事前にプリフェッチ命令を発行しておき、プリフェッチが完了するまでの間に別の演算を実行する。その後、ロード命令によりキャッシュ上のデータをアクセスする。この方法によりメモリアクセスによるレイテンシを隠すことができる。このようなプリフェッチの方法については、例えば「Todd C. Mowry 他: Design and Evaluation of Compiler Algorithm for Prefetching, Architectural Support for Programming Languages and Operating Systems, pp. 62-73, 1992」などに述べられている。

メモリアクセスのレイテンシを隠す別の機構として、プリロードと呼ばれる方法がある。これについては、「澤本他：RISC 超並列スーパーコンピュータのデータバス技術、情報処理 38 巻 6 号、pp. 485-492、1997 年」に述べられている。プリロード命令は、主記憶上のデータをキャッシュをバイパスして直接レジスタに書き込む。コンパイラでは、ソフトウェアパイプライン等のスケジューリング技法を用いて、プリロード命令とデータを使用する演算命令をメモリレイテンシ以上離すことにより、メモリレイテンシを隠す。

プリロードは、プリフェッチと比較して次のような利点がある。プリロード 1 命令で主記憶からレジスタへのロードができるので、プリフェッチのように命令を付加する必要がなく、発行命令数が増加しない。キャッシュへの書き込みがないため、メモリスループットがよい。また、プリフェッチはデータの使用前にキャッシュから追い出されることがあるが、プリロードは直接レジスタに書き込むのでその心配はない。

一方、プリフェッチには次の利点がある。プリフェッチ命令はレジスタを占有しないため、レジスタプレッシャを増大しない。またプリフェッチ命令は、1 度のメモリ要求で 1 キャッシュライン分をまとめてキャッシュに書き込み、連続データのアクセスではこれを有効利用できる。

前記の澤本らの文献にあげられているアーキテクチャは、プリフェッチとプリロードの両方の機構を備えており、プリロードは浮動小数点データに、プリフェッチは固定小数点と浮動小数点データの両方に適用できる。そこで固定小数点データはプリフェッチでキャッシュに、浮動小数点データはプリロードで直接レジスタにロードするコードを生成することが記載されている。1 つのループ内の浮動小数点データに対して、プリフェッチとプリロードの 2 つの方式を使い分けることに関しては述べられていない。

プリフェッチに関しては、各メモリアクセスに対してプリフェッチが必要かどうかを解析し、冗長なプリフェッチ命令を削除する方法が研究されている。これは、前記の Mowry らによる文献などに述べられている。この方法はループ

ネストのリユース解析に基づく。同じキャッシュラインのデータを2度以上参照するとき、リユースがあるという。リユースは、1つの参照が異なるループイタレーション間で同じキャッシュラインをアクセスする自己リユースと、複数の参照間の群リユースに分類される。リユース解析では、配列の添字式をループ制御変数の線形式として表現し解析する。リユース解析の方法については、「M. E. Wolf and M. S. Lam: A Data Locality Optimizing Algorithm, Programming Language Design and Implementation, pp.30-44, 1991」に詳しく述べられている。冗長なプリフェッチ命令削除では、群リユースに着目する。群リユースを持つ複数の参照の中で、最初に新たなデータを参照するものをリーディング参照とする。リーディング参照にはプリフェッチを適用し、他のデータはこのプリフェッチでキャッシュに書き込んだデータを利用するのでプリフェッチを省略する。このようにして必要なデータだけにプリフェッチ命令を出す。

ループ間のリユース解析を実施して、プリフェッチを削減する最適化については、「Keith Cooper 他: Cross-loop Reuse Analysis and its Application to Cache Optimizations, Workshop on Languages and Compilers for Parallel Computing, pp.1-15, 1996」に述べられている。本文献によれば、各ループ内の配列の参照部分を求め、データフローで伝播することによりループ入口およびループ出口に到達するデータ集合を求める。ループ入口と出口の両方に到達するデータはプリフェッチ不要とすることで、プリフェッチを削減する。

上記で述べたように、プリフェッチとプリロードにはそれぞれの利点・欠点がある。従来のように、全てのデータをプリフェッチ、または全てのデータをプリロードする方法では、欠点がでてしまう。そこで、各メモリ参照の特性に応じて、プリフェッチまたはプリロードのうちより適した方法を選択し、プリフェッチとプリロードを併用するコードを生成することによって、両方の方式の利点を活用することができる。

本発明の目的は、プリロードとプリフェッチの2つの方式を適用可能なメモ

リ参照を対象に、その参照により適したアクセス方法を選択し、プリロードとプリフェッチを併用したコードを生成することによって、より実行性能の高いコードを生成する最適化方法を提供することにある。

## 発明の開示

本発明の目的は、各メモリ参照に対してプリフェッチ、プリロード、又はロードの、どのアクセス方法を選択するかを決定するメモリアクセス方法判定ステップと、プリロードと判定されたメモリアクセスを対象に最適化を実施しプリロードコードを生成するプリロード最適化ステップと、プリフェッチと判定されたメモリアクセスを対象にプリフェッチコードを生成するプリフェッチ最適化ステップによって達成される。

メモリアクセス方法判定ステップの第1の方法は、そのメモリアクセスに対してソースプログラムの記述またはコンパイラオプションによるメモリアクセス方法の指定があるかどうかを解析するステップと、その結果に応じてメモリアクセス方法を決定するステップとから成る。

メモリアクセス方法判定ステップの第2の方法は、データが既にキャッシュにあるかどうかを判定するステップと、他のデータとのキャッシュ競合を判定するステップと、データが後で再参照されるかどうかを判定するステップと、レジスタ資源の制約を満たすかどうかを判定するステップとから成り、データが既にキャッシュにあるかどうか、およびデータが後で再参照されるかを判定するステップは、それぞれループ内に関する解析とループ間に関する解析から成る。

## 図面の簡単な説明

第1図は、本発明に係るメモリアクセス最適化方法を用いたコンパイラが稼動するシステムの構成図を示し、第2図は、本発明に係るメモリアクセス最適化方法を用いたコンパイラの処理手順を示し、第3図は、メモリアクセス方法

の指示を付加されたソースプログラムの一例であり、第4図は、コンパイラオプションによるメモリアクセス方法指示の一例であり、第5図は、第3図のプログラムに対する中間語の一例であり、第6図は、ループ表の一例であり、第7図は、アクセス方法登録表の一例であり、第8図は、本発明に係るメモリアクセス最適化方法におけるメモリアクセス方法判定の処理手順を示し、第9図は、本発明に係るメモリアクセス方法判定において、あるメモリアクセスの指定文中の指定の有無を解析する処理手順を示し、第10図は、本発明に係るメモリアクセス最適化方法におけるプリロード最適化の処理手順を示し、第11図は、第5図の中間語の基本ブロックB3に対応するDAGの例であり、第12図は、本発明に係るプリロード最適化におけるDAGエッジ上へのレイテンシ設定の処理手順を示し、第13図は、第5図の中間語に対してプリロード最適化を適用した後の中間語の一例であり、第14図は、本発明に係るメモリアクセス最適化方法におけるプリフェッチ最適化の処理手順を示し、第15図は、第13図の中間語に対してプリフェッチ最適化を適用した後の中間語の一例であり、第16図は、本発明に係るメモリアクセス最適化方法におけるコンパイラの解析によるメモリアクセス方法判定の処理手順を示し、第17図は、ループ表の一例であり、第18図は、本発明に係るメモリアクセス方法判定における先行ループの参照データ解析の処理手順を示し、第19図は、本発明に係るメモリアクセス方法判定における後続ループの参照データ解析の処理手順を示し、第20図は、リユース登録表の一例であり、第21図は、本発明に係るメモリアクセス方法判定におけるアクセス方法選択の処理手順を示し、第22図は、本発明に係るアクセス方法選択において、データがキャッシュ上にあるかを判定する処理手順を示し、第23図は、本発明に係るアクセス方法選択におけるキャッシュ競合解析の処理手順を示し、第24図は、本発明に係るアクセス方法選択において、データが後で再参照されるかを判定する処理手順を示し、第25図は、本発明に係るアクセス方法選択において、データが後で再参照されるかを判定する処理手順を示し、第26図は、本発明に係るアクセス方法選

択におけるレジスタ資源制約の判定の処理手順を示し、第 27 図は、ソースプログラムの一例であり、第 28 図は、ソースプログラムの一例である。

発明を実施するための最良の形態

以下、図面を用いて本発明の実施の形態について説明する。

本発明の実施の形態の第 1 の例は、ユーザ指示に従ったメモリアクセス方法判定によるメモリアクセス最適化である。

第 1 図は、本発明のメモリアクセス最適化方法を適用したコンパイラが稼動する計算機システムの構成図である。この計算機システムは、CPU 101、ディスプレイ装置 102、キーボード 103、主記憶装置 104、および外部記憶装置 105 より構成されている。キーボード 103 により、ユーザからのコンパイラ起動命令を受け付ける。コンパイラ終了メッセージやエラーメッセージは、ディスプレイ装置 102 に表示される。外部記憶装置 105 には、ソースプログラム 106 とオブジェクトプログラム 107 が格納される。主記憶装置 104 には、コンパイラ 108、コンパイル過程で必要となる中間語 109、ループ表 110、およびアクセス方法登録表 111 が格納される。リユース登録表 112 は第 2 の実施例のためのもので、本実施例では使用しない。コンパイル処理は CPU 101 によって制御され実行される。

第 2 図に、第 1 図のシステムで稼動するコンパイラ 108 の処理手順を示す。

コンパイラの処理は、構文解析 201、メモリアクセス方法判定 202、プリロード最適化 203、プリフェッチ最適化 204、コード生成 205 の順で行う。

構文解析 201 では、ソースプログラム 106 を入力として構文解析、及びループ解析を行い、中間コード 109 とループ表 110 を出力する。構文解析処理及びループ解析処理に関しては、例えば「エイホ、セシィ、ウルマン著：コンパイラ（サイエンス社、1990 年）」に記述されている。中間コード及びループ表については、後で説明する。



メモリアクセス方法判定 202 では、各メモリアクセスごとに適用すべきアクセス方法を判定し、その判定結果をアクセス方法登録表 110 に登録する。このメモリアクセス方法判定 202 は、本発明の特徴となる部分であるので、後に詳細に説明する。

プリロード最適化 203 では、アクセス方法判定 202 でプリロードと判定されたメモリアクセスを対象に、ソフトウェアパイプラインの技法を用いた命令スケジューリング最適化を実施し、プリロードコードを生成する。プリフェッチ最適化 204 では、アクセス方法判定 202 でプリフェッチと判定されたメモリアクセスを対象に、プリフェッチコードを生成する最適化を実施する。プリロード最適化及びプリフェッチ最適化については、後で第 10 図から第 15 図を用いて説明する。

コード生成 205 では、中間コード 109 をオブジェクトプログラム 107 に変換し、出力する。

第 3 図に、アクセス方法指示が付加されたソースプログラムの例を示す。指示文 301 はプリフェッチ指示であり、プリフェッチすべき配列名または配列要素をカンマで区切って指定している。指示文は直後のループネスト内のメモリアクセスに対して有効である。指示文 302 は、プリロード指示である。

ユーザ指示は、ユーザがコンパイラ起動時にコンパイラオプションによって指定することもできる。第 4 図は、コンパイラオプションによるユーザ指定の例を示す。オプション 401 は、プリフェッチ指示オプションであり、指示文 301 と同様の意味を持つ。オプションには対象ループと、配列名または配列要素を指定できる。対象ループ「loop1」は、プログラム内の 1 番目のループネストを表す。オプション 402 はプリロード指示文である。

第 5 図は本実施例におけるコンパイラの中間コードの例で、第 3 図のソースプログラムに対応している。中間コードは構文解析 201 で作成する。中間コードは、基本ブロックをエッジで結んだグラフで表現されている。このようなグラフは制御フローグラフと呼ばれている。B0 から B5 は、それぞれ基本ブ

ロックを表す。基本ブロックは、複数の文で構成される。各文は実行命令にほぼ対応しているので、「命令」と呼ぶこともある。「t1=load(a[i][j])」は、a[i][j]の値をロードして一時変数 t1 に代入するという意味である。本実施例で最適化の対象となるメモリアクセスは「load」で表現されている。

第6図は本実施例におけるループ表の例である。このループ表は、構文解析201で作成される。601はループ番号、602はループに属する基本ブロックを示す。各ループにつけられたユーザ指示は、プリフェッチ指示文603、及びプリロード指示文604に登録される。

第6図のループ表は、第3図のソースプログラム及び第5図の中間コードに対応している。ループ番号1は内側ループを、ループ番号2は外側ループを表している。プリフェッチ指示文及びプリロード指示文は、内側ループであるループ番号1のループに対応するものとして、登録される。

第7図にアクセス方法登録表の一例を示す。アクセス方法登録表は、メモリアクセスが属するループを示すループ番号701、メモリアクセス702、アクセス方法703のフィールドを持つ。これらのフィールドは、メモリアクセス方法判定202で設定され、プリロード最適化203、プリフェッチ最適化204は、登録されたアクセス方法に従った処理を行う。

次に、アクセス方法判定202の処理手順を詳細に説明する。

第8図に、ユーザ指示によるアクセス方法判定の処理手順を示す。各ステップを説明する。

ステップ801は、中間コード上に未処理のメモリアクセスがあるかどうかを判定し、なければ処理を終了する。あれば、ステップ802へ進み、未処理のメモリアクセスを取り出し、REFとする。ステップ803では、REFの属するループを求め、LOOPとする。各参照が属するループは、中間コードからその参照が属する基本ブロックを求め、ループ表110で基本ブロックを探すことにより求められる。

ステップ804では、ループ表110を参照し、LOOPへのプリフェッチ指定

文が登録されているかどうかを調べる。あればステップ 805 へ進みそのプリフェッチ指定文を取り出し、ステップ 806 で指定文中に REF の指定があるかどうかを判定する。ステップ 806 の詳細については第 9 図を用いて後述する。指定がある場合にはステップ 807 へ進み、アクセス方法にプリフェッチを選択する。

プリフェッチの指定がない場合にはステップ 808 へ進み、LOOP へのプリロード指定があるかどうかを調べる。ある場合にはステップ 809 でプリロード指定文を取り出し、ステップ 810 で指定文中に REF の指定があるかどうかを調べる。ステップ 810 の詳細については第 9 図を用いて後述する。あればステップ 811 に進み、プリロードを選択する。

プリロード指定がない場合にはステップ 812 へ進み、ロードを選択する。

ステップ 813 では、選択したアクセス方法をアクセス方法登録表に登録する。

ステップ 806、およびステップ 810 において、指定文中に REF の指定があるかどうかを調べる処理手順の詳細を示したものが第 9 図である。第 9 図の各ステップを説明する。

ステップ 901 は、指示文中に未処理の指定項目があるかどうかを判定する。ない場合はステップ 907 へ進み、REF の指定なしと判定して処理を終了する。ある場合にはステップ 902 へ進み、指定文から次の指定項目を取り出す。指定項目は配列名または配列参照を指定している。

ステップ 903 では配列要素の指定かどうか調べる。そうであればステップ 904 へ進み、配列要素が REF と一致するかどうか調べる。一致した場合はステップ 906 へ進み、REF の指定ありと判定して処理を終了する。一致しない場合には、ステップ 901 へ戻り、次の指定項目の処理へ進む。

ステップ 903 で配列要素指定でないと判定した場合にはステップ 905 へ進む。この場合は配列名の指定であるので、配列名が REF の配列と一致するかどうか調べる。一致した場合にはステップ 906 へ進み、REF の指定ありとして

処理を終了する。一致しない場合にはステップ901へ戻る。

以上で第1の実施例におけるメモリアクセス方法判定202の処理手順の説明を終了する。

次に、第8図および第9図の処理手順に従い、第3図のプログラムに対するアクセス方法判定処理を説明する。

ステップ801、ステップ802、ステップ803と進み、基本ブロックB3内のメモリアクセス $a[i][j]$ と、ループ1を取り出す。ステップ804、ステップ805と進み、ループ1へのプリフェッチ指定「a,  $c[i][j]$ 」を取り出す。ステップ806で、第9図に従い、指定文中に $a[i][j]$ の指定があるかを判定する。

ステップ901、ステップ902で、プリフェッチ指定の最初の指定項目「a」を取り出す。これは配列名の指定である。ステップ903で配列要素指定ではないのでステップ905へ進む。ステップ905で、REF「 $a[i][j]$ 」の配列名「a」と、指定項目「a」を比較し、一致するのでステップ906へ進み、指定ありとして終了する。

ステップ806で指定ありと判定されたので、ステップ807、ステップ813と進み、アクセス方法登録表に、「ループ1、 $a[i][j]$ 、プリフェッチ」を登録する。

次に、ステップ801へ戻り、次のメモリアクセス「 $b[j][i]$ 」について調べる。ステップ805でループ1のプリフェッチ指定「a,  $c[i][j]$ 」を取り出し、 $b[j][i]$ の指定があるか調べる。指定がないので、ステップ808、ステップ809と進み、プリロード指定「b」を取り出す。ステップ810で指定があると判定する。ステップ811、ステップ813と進み、アクセス方法登録表に、「ループ1、 $b[j][i]$ 、プリロード」を登録する。

次に、ステップ801へ戻り、次のメモリアクセス「 $c[i][j]$ 」について調べる。ステップ805でループ1のプリフェッチ指定「a,  $c[i][j]$ 」を取り出し、ステップ806で、第9図に従い判定する。

ステップ901、ステップ902で、最初の指定項目「a」を取り出す。配列名の指定であるのでステップ905へ進み、「c[i][j]」の配列名と一致しないので、ステップ901へ戻る。次に2番目の指定項目「c[i][j]」を取り出す。配列要素の指定であるのでステップ903からステップ904へと進む。配列要素が一致するので、ステップ906へ進み、指定ありとする。

ステップ806で指定ありと判定されたので、ステップ807、ステップ813と進み、アクセス方法登録表に、「ループ1、c[i][j]、プリフェッチ」を登録する。

次に、ステップ203のプリロード最適化について説明する。

プリロード最適化は、最内側ループを対象に、ループ単位に処理を行う。プリロード最適化は、基本的にはソフトウェアパイプラインングの技法に従った処理を行う。ソフトウェアパイプラインングについては、例えば、「M. Lam : Software Pipelining: An Effective Scheduling Technique for VLIW Machines, Proc. Of the SIGPLAN '88 Conference on Programming Language Design and Implementation, pp.318-328, 1988」に述べられている。第10図は、1つのループに対するプリロード最適化の処理手順を示している。以下、第5図の中間コードに対する処理を例にあげながら、各ステップについて順に説明する。

ステップ1001は、DAG(directed acyclic graph)を作成する。第11図にDAGの例を示す。これは、第5図の中間コードの基本ブロックB3に対応している。DAGの各ノードは、基本ブロック中の各文に対応している。ノード間のエッジは、実行順序の制約を表す。例えば、ノード1103のmul演算は、ノード1101及びノード1102のloadの結果をオペランドとするので、ノード1101とノード1102は、ノード1103よりも先に実行しなければならない。この関係をノード1101からノード1103へのエッジ、及びノード1102からノード1103へのエッジで表す。

ステップ1002は、ステップ1001で作成したDAGのエッジ上にレイテンシを設定する。本ステップの処理手順の詳細を第12図に示す。第12図の

各ステップについて説明する。

本処理は、DAG 上のエッジを辿り、順に処理する。ステップ 1 2 0 1 では未処理のエッジがあるかどうか判定し、なければ処理を終了する。あれば、ステップ 1 2 0 2 へ進み、未処理のエッジを取り出して、処理対象とする。

ステップ 1 2 0 3 は、エッジの始点ノードがメモリアクセスかどうか調べる。メモリアクセスでなければ、ステップ 1 2 0 7 へ進み、始点ノードの演算に対応するレイテンシの値をエッジ上に設定して、このエッジの処理を終了し、ステップ 1 2 0 1 へ戻る。ステップ 1 2 0 3 でメモリアクセスであれば、ステップ 1 2 0 4 へ進む。

ステップ 1 2 0 4 では、始点ノードのメモリアクセスのアクセス方法を、アクセス方法登録表 1 1 1 により調べる。アクセス方法がプリロードであればステップ 1 2 0 5 へ進み、主記憶アクセスのレイテンシを設定する。プリロードでなければステップ 1 2 0 6 へ進み、キャッシュアクセスのレイテンシを設定する。

第 5 図の中間コードの例では、第 1 1 図のようにレイテンシが設定される。演算 mul および add に対するレイテンシは 2 サイクルであるとする。また、メモリアクセス  $a[i][j]$  および  $c[i][j]$  は、アクセス方法がプリフェッチであるので、キャッシュアクセスのレイテンシを設定する。この値は 2 サイクルとする。また、メモリアクセス  $b[j][i]$  はアクセス方法がプリロードであるので、主記憶アクセスのレイテンシとして 1 0 サイクルを設定している。

ステップ 1 0 0 3 では、設定したレイテンシの値をもとに、DAG ノードをソフトウェアパイプラインのステージに分割する。本例では、DAG ノード 1 1 0 2 を始点とするエッジのレイテンシが 1 0 サイクルと大きいことから、ノード 1 1 0 2 を第 1 ステージとし、その他のノードを第 2 ステージとする。

ステップ 1 0 0 4 でソフトウェアパイプラインによる命令スケジューリングで、中間コードを変更する。

ステップ 1 0 0 5 で、プリロード参照の中間コードを「preload」に変更する。

第5図の中間コードに対してプリロード最適化を適用した結果を、第13図に示す。

基本ブロックB2には第1ステージのDAGノード1102に対応する命令が、プロローグコードとして挿入される。基本ブロックB3がカーネルコードである。基本ブロックB4には、第2ステージに対応する命令がエピローグコードとして挿入されている。

配列bをアクセスする3つの命令は「preload」に変更されている。

この例では、スライドウィンドウ機構を利用したコードを示している。スライドウィンドウ機構に関しては、「澤本他：RISC超並列スーパーコンピュータのデータバス技術、情報処理38巻6号pp.485-492、1997年」に述べられている。基本ブロックB3内では、2イタレーション後に使用するbの要素をプリロードする。B3内の「slide(1)」は、イタレーション毎に、レジスタウィンドウを1ずつスライドする。したがって、b[i][j]をオペランドとする演算は、レジスタfr1を使用することとし、プリロード命令では2つ先のレジスタであるfr3に値を書き込んでいる。

以上でプリロード最適化203の説明を終了する。

次に、プリフェッチ最適化204について説明する。

プリフェッチ最適化の処理手順を第14図に示す。各ステップについて説明する。

ステップ1401では何イタレーション先をプリフェッチするかを求め、その値をnとする。この値の求めかたについては、例えば、「Todd C. Mowry 他：Design and Evaluation of a Compiler Algorithm for Prefetching, ASPLOS V pp. 62-73, 1992」に記載されている。

ステップ1402では、ループ内のメモリ参照に対して、メモリアクセス方法登録表111によりアクセス方法がプリフェッチであるかどうかを調べ、未処理のプリフェッチ参照を取り出す。なければ終了する。

ステップ1403では、取り出したメモリアクセスに対して、nイタレーシ

ョン後にアクセスするアドレスのプリフェッチコードを作成する。ステップ1404では、ステップ1403で作成したプリフェッチコードをループ内に挿入し、ステップ1402へ戻る。

第13図の中間コードに対してプリフェッチ最適化を適用した結果を第15図に示す。ステップ1401では、nの値を2にしている。メモリアクセス「a[i][j]」はアクセス方法がプリフェッチであるから、ステップ1403において2イタレーション先のプリフェッチコードとして「prefetch a[i][j+2]」を作成する。ステップ1404でこのコードを中間コードに挿入する。第15図ではa[i][j]のloadの直前に挿入している。メモリアクセス「c[i][j]」に対しても同様にプリフェッチコードを作成し、挿入する。

以上で、本発明の実施形態の第1の例の説明を終了する。

次に、本発明の実施形態の第2の例として、コンパイラの解析によるアクセス方法判定について述べる。

第1図に示したコンパイラが稼動するシステムの例に関しては、第1の実施例と同様である。

コンパイラの処理手順は、第1の実施例と同様に第2図に従う。

ステップ201では構文解析を行い、中間コード及びループ表を作成する。中間コードの表現は、第1の実施例と同様である。ループ表に関しては、第17図を用いて後述する。リユース登録表112は、メモリアクセス方法判定202で使用する。これについては後で第20図を用いて説明する。

ステップ202では、中間語およびループ表の情報を解析し、メモリアクセス方法を判定してアクセス方法登録表に登録する。アクセス方法登録表は、第1の実施例と同様で、第7図のようである。メモリアクセス方法判定の処理手順については、本実施例の特徴となる部分であるので、後で詳細に説明する。

ステップ203では、メモリアクセス方法登録表にプリロードと登録されたメモリアクセスを対象とした最適化を行う。ステップ204では、プリフェッチと登録されたメモリアクセスを対象とした最適化を行う。ステップ205で



は、これらの最適化を施した中間語を対象にコード生成を実施し、オブジェクトコードを生成する。これらのステップについては第 1 の実施例と同様であるので、説明を省略する。

以下、本実施例におけるメモリアクセス方法判定 202 に関して、詳細に説明する。

メモリアクセス方法判定は、ループ内の解析と、ループ間の解析を含んでいる。以下では、ループ内及びループ間の解析を両方実施する処理手順について説明するが、一方を省略して判定することも可能である。

第 16 図は、本実施例におけるアクセス方法判定の処理手順を示す。以下、各ステップについて説明する。

ステップ 1601 では、ループの処理順序を決める。なるべく実行順にループを処理するように順序付ける。処理対象のループは最内側ループとする。順序付けた結果は、ループ表に登録される。本ステップは、ループ間の解析に係る。ループ内の解析だけを実施する場合には、任意の順序でよい。

ループ表を第 17 図に示す。1701 はループ番号を示す。1702 はループに属する基本ブロックを示す。各メモリアクセスが属するループを調べるには、属する基本ブロックを探せばよい。1703 は前のループ番号、1704 は次のループ番号を設定する。これらのフィールドにはループを順序付けた結果が設定される。これらのフィールドは、最内側ループに対して設定される。フィールド 1705 から 1708 はループ制御変数に関する情報である。これらはリユース解析などで配列添字を解析する際に利用される。

ステップ 1602 では、ステップ 1601 で決めたループの処理順序に従い、次の処理ループを取り出す。次のループがなければ処理を終了する。

ステップ 1603 はループ間の解析に係り、先行ループの参照データを解析する。先行ループとは、現在の処理ループよりも前に実行されるループを指す。ループの処理順序が実行順になっていれば、先行ループは既にアクセス方法判定の処理済みであるので、先行ループ内の参照のアクセス方法を用いた解

析を行う。

本ステップの詳細な処理手順を示したものが、第18図である。各ステップについて説明する。

ステップ1801では、前のループがあるかどうかを判定する。前のループはループ表の前ループフィールド1703に従って辿る。前のループがあればそれを取り出してステップ1802へ進み、なければ処理を終了する。

ステップ1802では、取り出した前のループ内に未処理のメモリアクセスがあるかどうかを調べる。あればステップ1803へ進み、なければステップ1806へ進む。ステップ1803では、メモリアクセスのアクセス方法を、アクセス方法登録表により調べ、プリフェッチまたはロードならばステップ1804へ進む。そうでない場合、即ちプリロードの場合には、ステップ1802へ戻り、次のメモリアクセスを処理する。

ステップ1804およびステップ1805は、プリフェッチまたはロード、即ちキャッシュをアクセスするメモリアクセスに関する処理である。ステップ1804では、本メモリアクセスのループでの参照範囲を求める。ステップ1805では、その参照範囲を先行ループの新参照データ集合に加える。例えば、メモリアクセス $a[i]$ があり、ループ制御変数 $i$ の範囲が $1 \leq i \leq 100$ であるとき、 $a[1:100]$ を新参照データ集合に加える。

ステップ1806では、新参照データ集合がキャッシュ容量以下かどうかを調べる。キャッシュ容量以下であればステップ1807へ進み、参照データ集合に新参照データ集合を設定した後、ステップ1801へ戻り、次の前ループを処理する。新参照データ集合がキャッシュ容量を越えていれば、処理を終了する。これは、前のループをこれ以上調べても、現在のアクセス方法判定の対象ループの入口では、キャッシュから追い出されてしまっているからである。この場合、最後に処理したループの参照データは、参照データ集合には加えられていない。

ステップ1604もループ間の解析に関係し、後続ループの参照データを解

析する。後続ループとは、現在の処理ループよりも後に実行されるループを指す。本ステップの詳細な処理手順を第19図に沿って説明する。

ステップ1901では、ループ表の次ループフィールド1704に従い、次のループを取り出す。次のループがなければ処理を終了する。

ステップ1902では取り出したループ内に未処理のメモリアクセスがあるかを調べる。あればステップ1903へ進み、なければステップ1905へ進む。

ステップ1903では、メモリアクセスのループでの参照範囲を求め、ステップ1904ではその参照範囲を後続ループの新参照データ集合に加える。

ステップ1905では、新参照データ集合がキャッシュ容量以下かどうかを調べ、そうであれば、参照データ集合に新参照データ集合を設定した後、ステップ1901へ戻り、そうでなければ処理を終了する。

ステップ1605は、ループ内の解析に関係する。本ステップではループ内データのリユース解析を実施し、その結果をリユース登録表112に登録する。

リユース登録表を第20図に示す。フィールド2001には、リユース解析の対象としたメモリアクセスに登録する。リユース解析の結果は自己リユース、群リユースに分けて登録される。自己リユースとは、1つのメモリアクセスが何度も参照されるときに同じキャッシュラインをアクセスすることによるリユース、群リユースとは、複数のメモリアクセス間で同じキャッシュラインを参照することによるリユースである。フィールド2002には、自己リユースの有無に登録する。フィールド2003には群リユースの情報を登録する。群リユースを持つメモリアクセスを集合にまとめ、リユースをもつ参照間で最初に参照されるメモリアクセスを選んでリーディング参照とする。これらを、フィールド2004、2005、2006から成る表に登録する。フィールド2003には、そのメモリアクセスが属する群リユースの番号に登録する。リユース解析の方法については、公知の技術であるので、詳しい説明は省略する。

ステップ1606からステップ1609は、ループ内の各メモリアクセスに

対する処理である。ステップ1606ではループ内に未処理のメモリアクセスがあるかを判定し、なければステップ1602に戻り、次のループの処理を行う。未処理のメモリアクセスがあれば、ステップ1607へ進み、メモリアクセスを取り出して処理対象とする。ステップ1608では、取り出したメモリアクセスのアクセス方法を選択する。ステップ1609では、ステップ1608で選択したアクセス方法を、アクセス方法登録表111に設定し、ステップ1606に戻って次のメモリアクセスを処理する。

ステップ1608のアクセス方法選択処理の詳細を示しているのが第21図である。この処理手順は、本実施例の最も特徴的な部分である。各ステップについて順に説明する。

ステップ2101、2102、および2103は、キャッシュに関連した判定を行う。ステップ2101では、メモリアクセスの対象データがすでにキャッシュ上にあるかどうかを判定する。あればステップ2107へ進みロードを選択する。なければステップ2102へ進む。ステップ2102では、処理対象のメモリアクセスが、他のキャッシュデータとキャッシュ競合するかどうかを判定する。競合する場合にはステップ2106へ進み、プリロードを選択する。競合しない場合にはステップ2103へ進む。ステップ2103では、処理対象のメモリアクセスと同一キャッシュラインのデータを後で再び参照するかどうかを調べる。そうであればステップ2105へ進み、プリフェッチを選択する。そう出なければステップ2104へ進む。

ステップ2104はレジスタ資源に関する判定を行う。プリロードを適用するメモリアクセスは長時間にわたってレジスタを占有するため、プリロードの過剰使用によるレジスタ不足を防ぐことが目的である。レジスタ資源の制約条件を満たせばステップ2106へ進み、プリロードを選択する。満たさなければ、ステップ2105へ進み、プリフェッチを選択する。

ステップ2101、2102、2103、および2104の各条件判定の詳細については、以下で説明する。

ステップ2101の処理手順の詳細を示しているのが、第22図である。各ステップについて説明する。

ステップ2201では、ステップ1603で解析した先行ループの参照データ集合を参照し、現在処理中のメモリアクセスが集合内に含まれているかどうかを判定する。含まれている場合には、先行ループでキャッシュに書き込まれているので、ステップ2204へ進み、キャッシュ上にありと判定して処理を終了する。先行ループの参照データ集合に含まれていない場合には、ステップ2202へ進む。

ステップ2202では、ステップ1605の解析により作成したリユース登録表を参照し、群リユースがあるかどうかを判定する。ない場合にはステップ2205へ進み、キャッシュ上にないと判定する。群リユースがあればステップ2203へ進み、群リユースのリーディング参照であるかどうかを判定する。群リユースは、リーディング参照の参照時に主記憶からキャッシュへの書き込みを行い、残りの参照はキャッシュを再利用する。従って、リーディング参照であるときは、ステップ2205へ進みキャッシュ上にないと判定する。そうでないときは、ステップ2204へ進み、キャッシュ上にあると判定する。

ステップ2201はループ間の解析に関するもので、ループ入口でデータがキャッシュにのっているかどうかを調べている。ステップ2202および2203は、ループ内の解析に関するもので、ループ開始からアクセス直前までの間に、他のメモリアクセスにより参照データがキャッシュに書き込まれるかどうかを調べている。

第23図は、ステップ2102のキャッシュ競合判定の処理手順を示している。本解析はループ内の他の参照との間のキャッシュ競合を調べる。

ステップ2301では、アクセス方法登録表に同一ループに属する参照が登録されているかを調べ、あれば取り出してステップ2302へ進む。未処理の参照がなければステップ2306へ進み、キャッシュ競合しないと判定して処理を終了する。

ステップ2302では、取り出した参照のアクセス方法がプリフェッチまたはロードであるかを調べる。そうであればキャッシュに書き込まれるのでステップ2303へ進み、キャッシュ競合を調べる。そうでなければアクセス方法はプリロードであり、キャッシュへ書き込まれないので、ステップ2301へ戻る。

ステップ2303では、現在処理中の参照と、ステップ2301で取り出した参照の間のアドレス差を計算する。ステップ2304では、このアドレス差をキャッシュサイズで割った剰余が一定値以下であるかどうかを判定する。一定値以下であればステップ2305へ進み、キャッシュ競合すると判定して終了する。一定値以下でなければこの2つの参照はキャッシュ競合しないので、ステップ2301へ戻り、次の参照を調べる。

次に、ステップ2103の処理の詳細を説明する。

初めに、簡易的な判定方法として、各メモリアクセスの添字の解析による判定について述べる。第24図に処理手順を示す。

ステップ2401では、配列添字を解析してストライドを計算する。ストライドは連続するイタレーションの参照間が配列要素いくつ分であることを示す。配列添字がループ制御変数の1次式で表現されるときにストライドは一定で、ループ制御変数の増分値と添字式におけるループ制御変数の係数との積がストライド値である。

ステップ2402では、ストライドが定数であるかどうかを判定する。定数でなければステップ2405へ進み、再参照しないと判定して処理を終了する。定数であればステップ2403へ進む。

ステップ2403では、ストライド判定の基準値を計算する。1つのキャッシュラインを $n$ 回アクセスするためのストライド値として、キャッシュラインサイズを配列の要素サイズと $n$ の積で割った値を計算し、これを $s$ とする。

ステップ2403ではストライドの値が基準値 $s$ 以下であるかどうかを判定する。 $s$ 以下であればキャッシュラインを $n$ 回以上アクセスするので、ステッ

プ2405へ進み再参照すると判定して終了する。そうでなければステップ2406へ進み、再参照しないと判定する。

次に、ステップ2103の別の処理手順として、ステップ1604および1605の解析結果を用いる方法について説明する。第25図に処理手順を示す。

ステップ2501では、リユース登録表を参照し、自己リユースがあるかどうかを判定する。あればステップ2506へ進み、再参照すると判定する。なければステップ2502へ進む。

ステップ2502では、リユース登録表を参照し、群リユースがあるかどうかを判定する。ある場合にはステップ2503へ進み、リーディング参照であるかどうかを判定する。リーディング参照である場合には同じ群の他の参照によるリユースがあるので、ステップ2506へ進み、再参照すると判定する。そうでない場合にはステップ2504へ進む。

ステップ2504はループ間の再参照を調べるステップである。ステップ1604で解析した後続ループの参照データを解析し、処理対象のメモリアクセスが含まれるかどうかを判定する。含まれればステップ2506へ進み、再参照すると判定する。含まれなければステップ2505へ進み、再参照しないと判定する。

以上で、第24図、第25図、およびステップ2103の処理手順の説明を終了する。

次に、ステップ2104のレジスタ資源の制約の判定について説明する。本ステップの処理手順の詳細を第26図に示す。

ステップ2601はアクセス方法登録表を参照し、同じループのメモリアクセスで既にプリロードと決定したものの数を数える。

ステップ2602は、プリロード数がレジスタ資源の制約を満たすかどうかを判定するステップである。最適なレジスタ資源の制約条件はスケジューリングの方法などに依存するが、ここではレジスタ数に一定の係数 $k$ を掛けた値をプリロード数の上限とすることにする。例えば、レジスタ数を32本、係数を0.5

とする場合、プリロード数が16を越えないようにする。 $(\text{プリロード数}) + 1 \leq (\text{レジスタ数}) * k$ であれば、ステップ2603へ進み、レジスタ資源の制約を満たすと判定する。そうでなければステップ2604へ進み、レジスタ資源の制約を満たさないと判定する。

以上で第2の実施例の処理手順の説明を全て終了する。

次にプログラムの例を用いて、本実施例による処理過程を説明する。

初めに単一ループのプログラムを例に説明する。第27図はソースプログラムの例である。

第16図の処理手順に従って、本プログラムのアクセス方法判定を実施する。

ステップ1601、1602、1603、1604はループ間の処理に関するものであるので、処理は不要である。

ステップ1605では、リユース解析を行い、第20図に示した結果が得られる。メモリアクセス「 $c[x[i]]$ 」は、ストライドが一定でないランダム参照であるので、自己リユースを持たない。その他のメモリアクセスは、連続参照またはストライド2の参照であり、連続するイタレーション間で同じキャッシュラインをアクセスするので、自己リユースを持つ。群リユースは同じ配列の参照に対して解析される。配列「a」は、2つの参照「 $a[i]$ 」「 $a[i+2]$ 」を持つ。これらは同じイタレーション内で同じキャッシュラインをアクセスする可能性があるため、これらを群番号1として登録する。リーディング参照は「 $a[i]$ 」とする。配列「b」も2つの参照を持つが、2つの参照間の距離が大きいため群リユースはなしと判定される。配列「d」の2つの参照は群リユースを持つ。「 $d[i+100]$ 」でアクセスした要素を、100イタレーション後に「 $d[i]$ 」がアクセスする。従って、「 $d[i+100]$ 」をリーディング参照として、この群リユースを群番号2として登録する。

次に、各メモリアクセスに対するアクセス方法の判定（ステップ1608）について説明する。初めに「 $a[i]$ 」のアクセス方法を判定する。ステップ2101では第22図に従った判定を実施する。先行ループはなく、群リユースの



リーディング参照であることから、キャッシュ上になしと判定される。次にステップ 2102 へ進み、第 23 図に従ってキャッシュ競合の解析を行う。ステップ 2301 で、アクセス方法登録表には他の参照は未登録であるので、ステップ 2306 へ進みキャッシュ競合しないと判定する。次にステップ 2103 で同一キャッシュラインを後で再参照するかを判定する。第 24 図については省略し、第 25 図の処理手順に従うものとする。ステップ 2501 で自己リユースがあるので、ステップ 2506 へ進み、再参照すると判定する。従って第 21 図の処理ではステップ 2105 へ進み、プリフェッチを選択する。次に「 $a[i+2]$ 」のアクセス方法を判定する。ステップ 2101 では、第 22 図の処理手順に従い、群リユースがあってリーディング参照でないことから、ステップ 2204 でキャッシュ上にありと判定する。従ってステップ 2107 へ進みロードを選択する。次に配列  $b$  について説明する。「 $b[2*i]$ 」は自己リユースがあることからプリフェッチを選択する。「 $b[2*i+1024]$ 」のステップ 2102 の処理について説明する。ステップ 2301 で、アクセス方法登録表にプリフェッチと登録された参照「 $b[2*i]$ 」を取り出す。2 データ間のアドレスは、 $8\text{byte} \times 1024 = 8\text{Kbyte}$  である。キャッシュサイズを  $4\text{Kbyte}$  とすると、ステップ 2304 の判定により、2 つのデータはキャッシュ競合すると判定される。2 つともプリフェッチを適用してしまうと、一方をプリフェッチした後、他方のプリフェッチによりキャッシュから追い出され、プリフェッチの効果がなくなってしまう。このような場合には 2 つ目の参照はプリフェッチを選択せず、ステップ 2106 へ進んでプリロードを選択する。他の参照についての説明は省略する。このようにしてメモリアクセス方法を判定した結果は、 $a[i]$ 、 $b[i*2]$ 、 $x[i]$ 、および  $d[i+100]$  がプリフェッチ、 $a[i+2]$  および  $d[i]$  がロード、 $c[x[i]]$  および  $b[i*2+1024]$  がプリロードとなる。

次に複数のループを含むプログラムの例を説明する。第 28 図にソースプログラムを示す。

ステップ 1601 で、ループの処理順序を決める。処理順序は実行順序に従

うことが望ましいので、LOOP1, LOOP2, LOOP3, LOOP4 の順に処理を行う。ループ表は、第17図のように登録されている。以下、「LOOP2」とコメント付けされた2番目のループの処理手順を説明する。

ステップ1603で先行ループの参照データを第18図の処理手順に従って解析する。ステップ1801で前のループとして LOOP1 を取り出す。ステップ1802でメモリアクセス「a[i]」を取り出す。LOOP1 の「a[i]」のアクセス方法はプリフェッチが指定されているものとする。ステップ1804で参照範囲を求める。制御変数 i は下限値が 0、上限値が 99、増分値が 1 であるので、参照範囲は a[0:99] となる。ステップ1805で新参照データ集合は {a[0:99]} となり、ステップ1802に戻って他のメモリアクセスがないのでステップ1806へ進む。新参照データ集合で参照するデータの容量は、8byte×100=800byte である。本例では、キャッシュサイズを 4Kbyte と仮定する。新参照データ集合はキャッシュ容量以下であるので、ステップ1807で参照データ集合を {a[0:99]} とする。ステップ1801へ戻り、LOOP1 の前ループがないので処理を終了する。

次に、ステップ1604で後続ループの参照データを解析する。ステップ1901で次のループとして LOOP3 を取り出す。メモリアクセス「b[i]」を解析し、新参照データ集合は {b[0:399]} となる。この集合は 8byte×400=3200byte 参照し、キャッシュ容量以下であるので、これを参照データ集合とし、次に LOOP4 を調べる。メモリアクセス「c[i]」の参照範囲を新参照データ集合に加えると、{b[0:399], c[0:399]} となる。この集合は 6400byte を参照し、キャッシュサイズの 4Kbyte を越えているので、処理を終了する。この結果、後続ループの参照データ集合として {b[0:399]} が得られる。

次に、ステップ1605でループ内のメモリアクセス a[i], b[i\*4], c[i\*4] についてリユース解析を行う。群リユースは存在しない。a[i] は自己リユースを持つ。キャッシュラインサイズを 32byte とすると、b および c は 1 ラインを 1 回しかアクセスしないので、自己リユースをもたない。

次に各メモリアクセスのアクセス方法を選択する。最初に  $a[i]$  について調べる。ステップ 2101 でキャッシュ上にあるかを調べる。ステップ 2201 で、 $a[i]$  の参照範囲  $a[0:99]$  は先行ループの参照データ集合に含まれるので、キャッシュ上にあると判定され、アクセス方法としてロードを選択する。次に、 $b[i*4]$  について調べる。ステップ 2101 で、 $b$  は先行ループの参照データ集合に含まれておらず、群リユースも持たないので、キャッシュ上にないと判定される。ステップ 2102 ではキャッシュ競合はないとする。ステップ 2103 で同一キャッシュラインデータを後で再参照するかを第 25 図に従い調べる。リユースを持たないので、ステップ 2501 およびステップ 2502 の判定を経てステップ 2504 へ進む。 $b[i*4]$  の参照範囲は  $b[0:396]$  で後続ループの参照データ集合に含まれるので、再参照すると判定する。ステップ 2105 へ進み、プリフェッチを選択する。次に、 $c[i*4]$  について調べる。 $c$  は先行ループの参照データ集合に含まれておらず、群リユースを持たないので、キャッシュ上にはないと判定される。キャッシュ競合もないとする。第 25 図の処理では、リユースを持たず、後続ループの参照データにも含まれないことから再参照しないと判定する。ステップ 2104 へ進み、レジスタ資源の制約を調べる。レジスタ数を 32、係数を 0.5 とすると、ループ内にプリロードと登録済みの他のメモリアクセスはないので制約を満たし、ステップ 2106 でプリロードを選択する。

以上で第 28 図のプログラムに対するメモリアクセス方法判定の処理手順の説明を終了し、第 2 の実施例の説明を終える。

#### 産業上の利用可能性

本発明によれば、ループ内の各メモリアクセスに対して、プリフェッチ、プリロード、またはロードなどの、そのメモリアクセスにより適したアクセス方法を選択することができる。その結果、プリロードとプリフェッチを併用したコードを生成することができ、より実行性能の高いオブジェクトコードを生成できる。

## 請 求 の 範 囲

1. 主記憶上のデータをキャッシュ上に書き込むプリフェッチ機構と、主記憶上のデータをキャッシュ上には書き込まずにレジスタに直接ロードするプリロード機構とを備えたアーキテクチャにおいて、

プリフェッチとプリロード両方の機構を利用したコードを出力するメモリアクセス最適化方法であって、

プログラム上の各メモリアクセスに対して、キャッシュから直接レジスタへ書き込むロード、またはキャッシュへの書き込み後にキャッシュからロードするプリフェッチ、またはプリロードのどれを適用すべきかを判定するアクセス方法判定ステップと、

前記アクセス方法判定ステップでプリロードと判定された参照に対してプリロードコードを生成するステップと、

前記アクセス方法判定ステップでプリフェッチと判定された参照に対してプリフェッチコードを生成するステップとを有し、

前記アクセス方法判定ステップは、

前記プログラム中のアクセス方法の指定を解析するステップを有し、

前記解析によりプリフェッチまたはプリロードと指定された参照に対しては、前記指定に従ってアクセス方法を判定することを特徴とするメモリアクセス最適化方法。

2. 前記アクセス方法の指定は、

プログラムソース内またはコンパイラコマンドラインに、ループ単位で指定し、プリフェッチまたはプリロードすべき配列名または配列要素を参照として指定することを特徴とする請求の範囲第1項記載のメモリアクセス最適化方法。

3. 主記憶上のデータをキャッシュ上に書き込むプリフェッチ機構と、主記憶上のデータをキャッシュ上には書き込まずにレジスタに直接ロードするプリロード機構とを備えたアーキテクチャにおいて、

プリフェッチとプリロード両方の機構を利用したコードを出力するメモリアクセス最適化方法であって、

プログラム上の各メモリアクセスに対して、ロード、またはプリフェッチ、またはプリロードのどれを適用すべきかを判定するアクセス方法判定ステップと、

アクセス方法判定ステップでプリロードと判定された参照に対してプリロードコードを生成するステップと、

アクセス方法判定ステップでプリフェッチと判定された参照に対してプリフェッチコードを生成するステップとを有し、

前記アクセス方法判定ステップは、

キャッシュまたはレジスタ資源の利用状況に基づいた判定を実施することを特徴とするメモリアクセス最適化方法。

4. 前記アクセス方法判定ステップは、

連続するイタレーション間でのメモリアクセスの参照距離が何要素分かというストライドを判定するステップと、

アクセス方法判定のためのストライド値の基準である基準ストライド値を決定するステップとを有し、

ストライドが基準ストライド値以下の参照に対しては、プリフェッチを選択し、その他の参照にはプリロードを選択して判定することを特徴とする請求の範囲第3項記載のメモリアクセス最適化方法。

5. 前記基準ストライド値を決定するステップは、

1回のプリフェッチでキャッシュに書き込まれるデータサイズと、1回のメモリアクセスが参照するデータサイズとに基づき決定することを特徴とする請求の範囲第4項記載のメモリアクセス最適化方法。

6. 前記アクセス方法判定ステップは、

メモリアクセス間での同一キャッシュラインの再参照を解析するステップを有し、

前に実行されるデータとの間に再参照があればロードを選択し、  
後に実行されるデータとの間に再参照があればプリフェッチを選択し、  
再参照がなければプリロードを選択して判定することを特徴とする請求の範囲第3項記載のメモリアクセス最適化方法。

7. 前記アクセス方法判定ステップは、

メモリアクセス間でのキャッシュ競合を解析するステップを有し、  
アクセス方法としてプリフェッチを選択した他のデータとキャッシュ競合を起こすデータに対しては、プリロードを選択して判定することを特徴とする請求の範囲第3項記載のメモリアクセス最適化方法。

8. 前記アクセス方法判定ステップは、

アクセス方法にプリロードを選択したループ内のメモリアクセスに対してレジスタの占有状況を判定するステップを有し、

レジスタの占有状況が所定の条件を満たしている場合には、プリフェッチを選択して判定することを特徴とする請求の範囲第3項記載のメモリアクセス最適化方法。

9. 前記メモリアクセス間での同一キャッシュラインの再参照を解析するステップは、

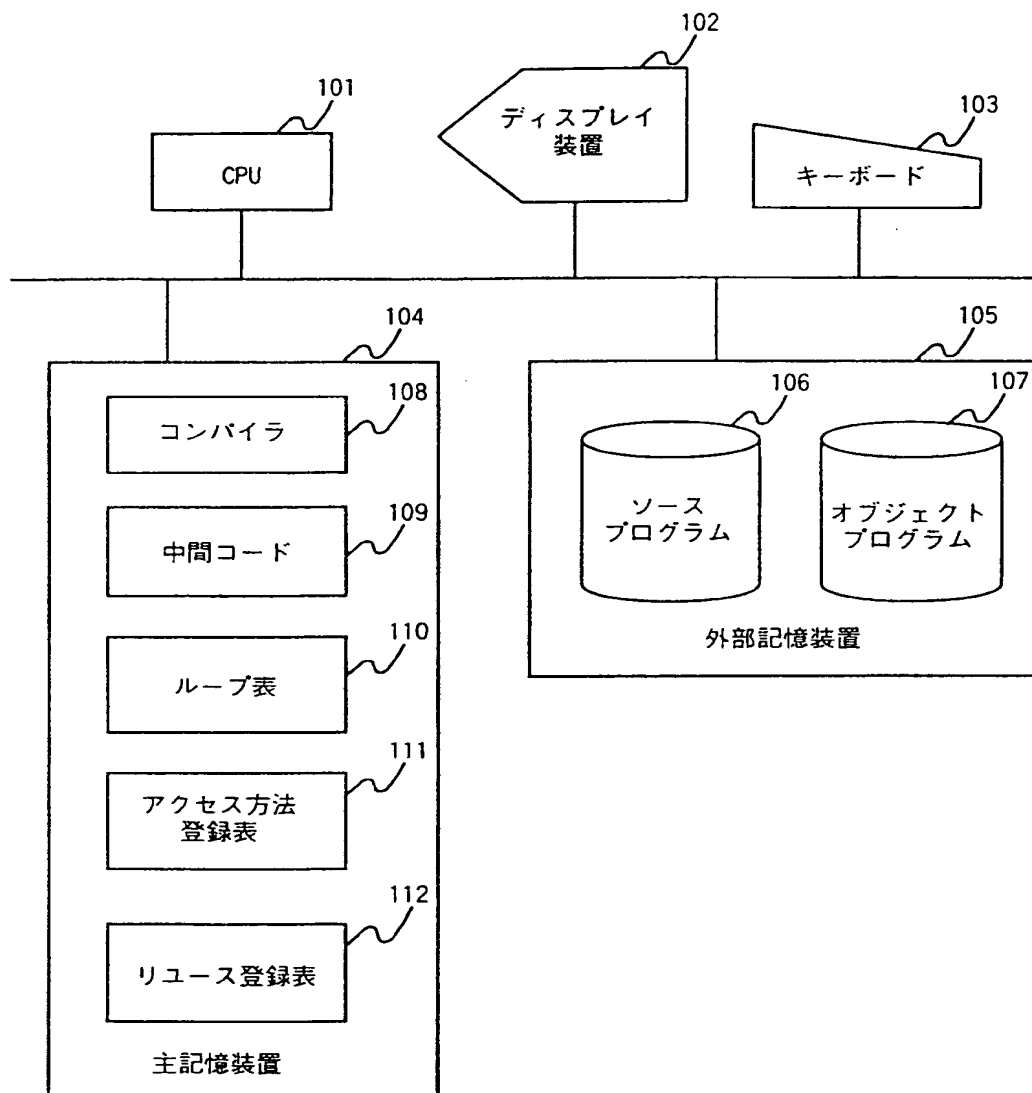
同じループに属する参照間での再参照を解析するステップと、  
異なるループ間の参照間での再参照を解析するステップとから成り、  
上記異なるループ間の再参照を解析するステップは、さらに、  
前に実行されるループ内のプリフェッチまたはロードを選択したメモリアクセスの参照範囲の和を解析するステップと、

後に実行されるループ内のメモリアクセスの参照範囲の和を解析するステップを有する

ことを特徴とする請求の範囲第6項記載のメモリアクセス最適化方法。

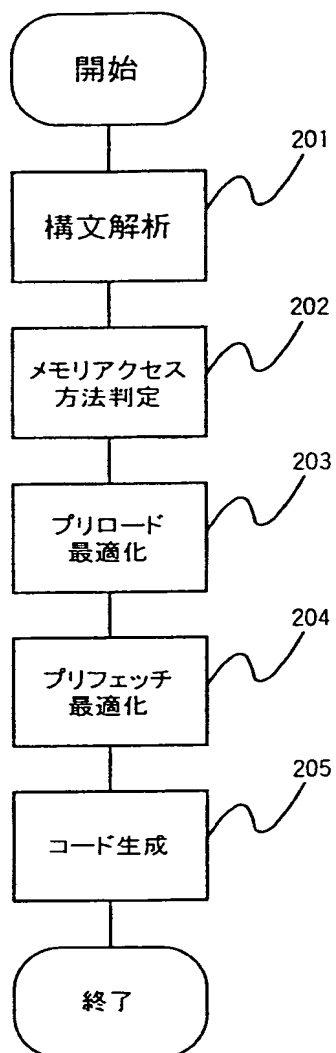
1 / 2 6

第 1 図



2 / 2 6

第 2 図





3 / 26

## 第3図

```
double a[100][100], b[100][100], c[100][100], s;  
void func()  
{  
    int    i, j;  
  
    #prefetch  a, c[i][j] 301  
    #preload   b           302  
  
    for (i = 0; i < 100; i++) {  
        for (j = 0; j < 100; j++) {  
            s = s + a[i][j] * b[j][i] + c[i][j];  
        }  
    }  
}
```

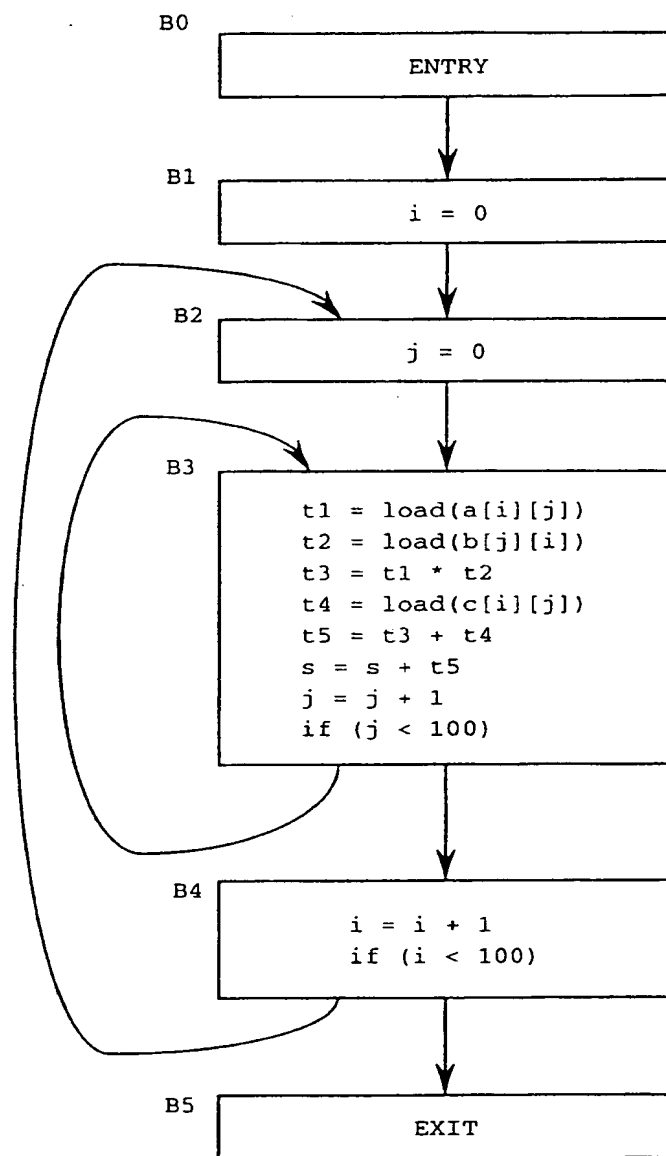
## 第4図

```
cc -Oprefetch,loop1,a,c[i][j] -Opreload,loop1,b func.c
```


401 402

4 / 2 6

第5図

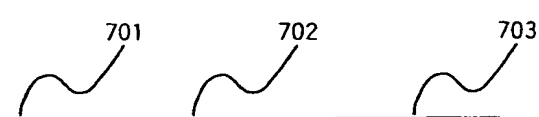


第6図



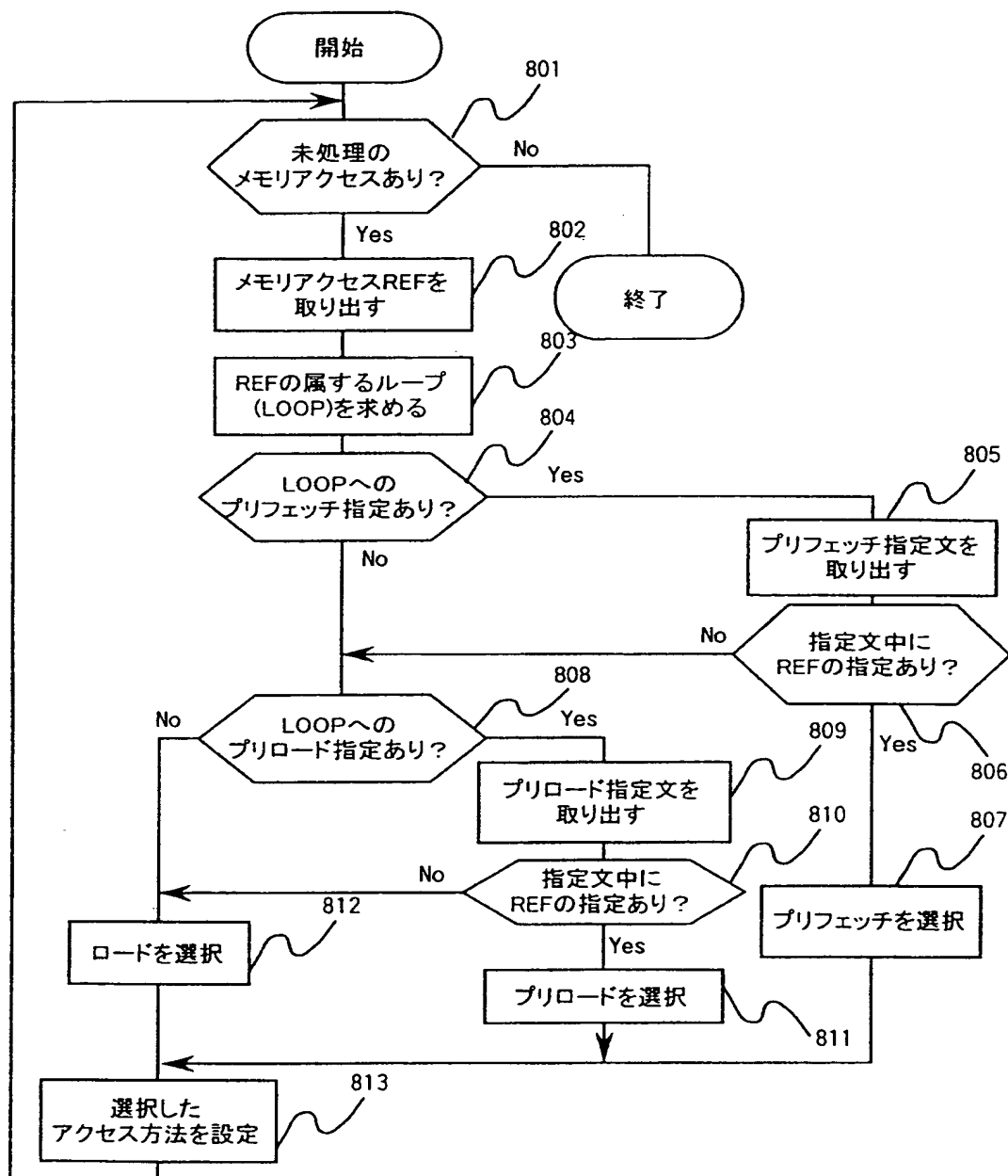
ループ番号	基本ブロック	プリフェッチ指示文	プリロード指示文
1	BB3	a, c[i][j]	b
2	BB2, BB4	-	-

第7図



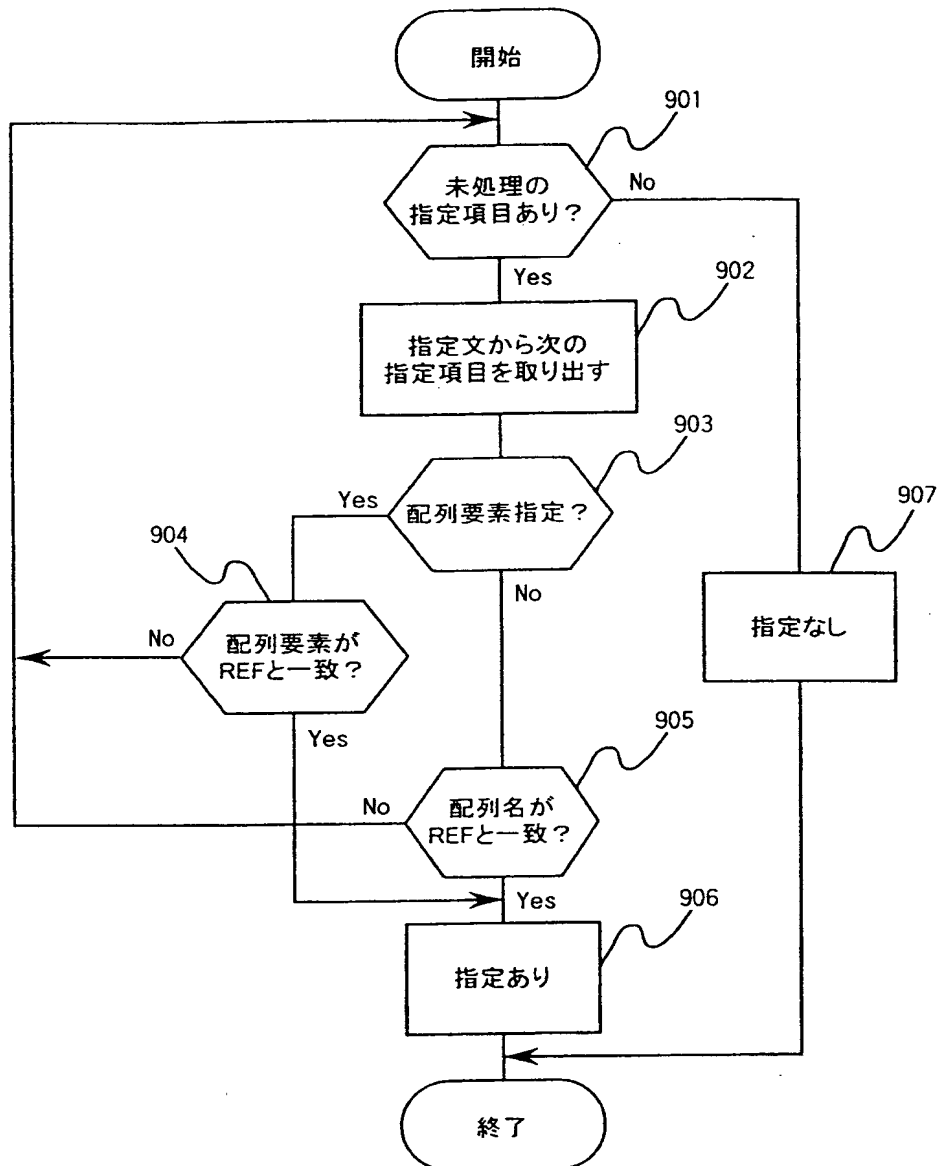
ループ	メモリアクセス	アクセス方法
1	a[i]	プリフェッチ
1	a[j]	プリロード
1	a[i+1]	ロード
2	b[j]	プリロード
3	c[k]	プリロード

第 8 図



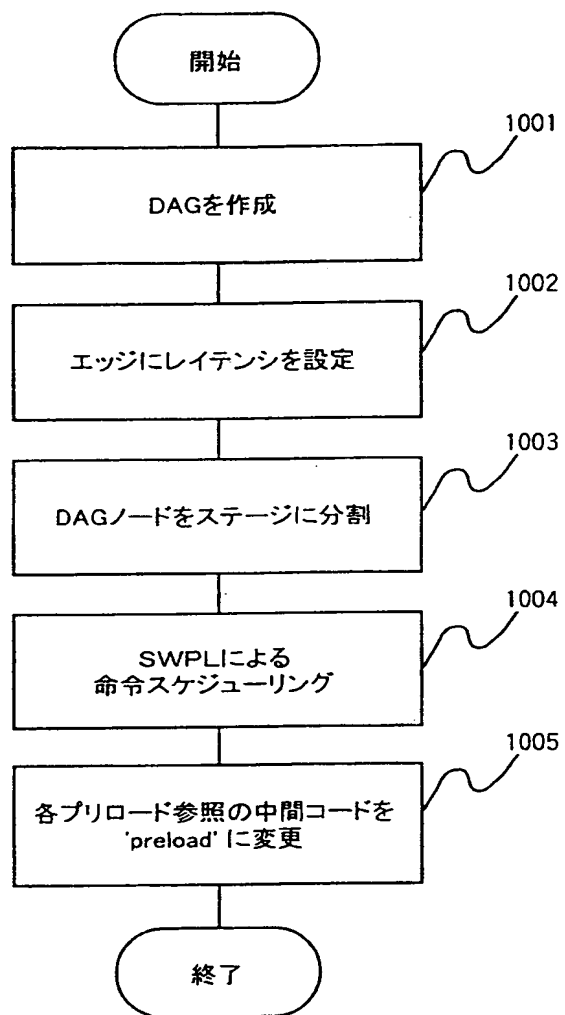
7 / 2 6

第 9 図

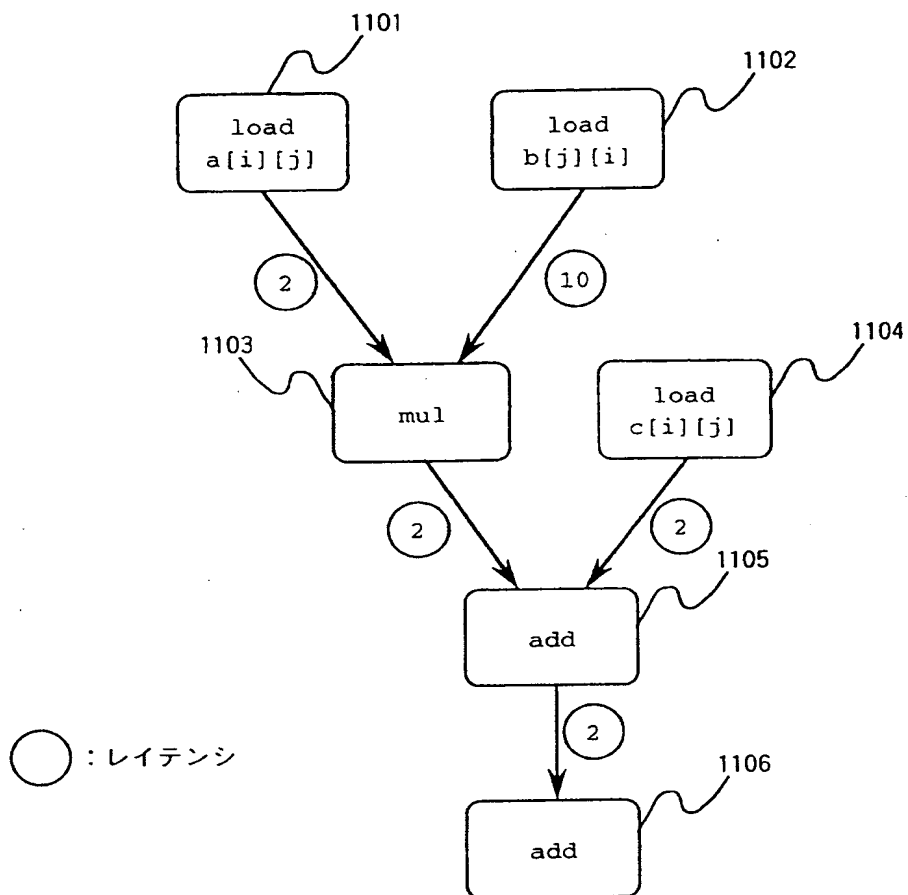


8 / 26

第10図

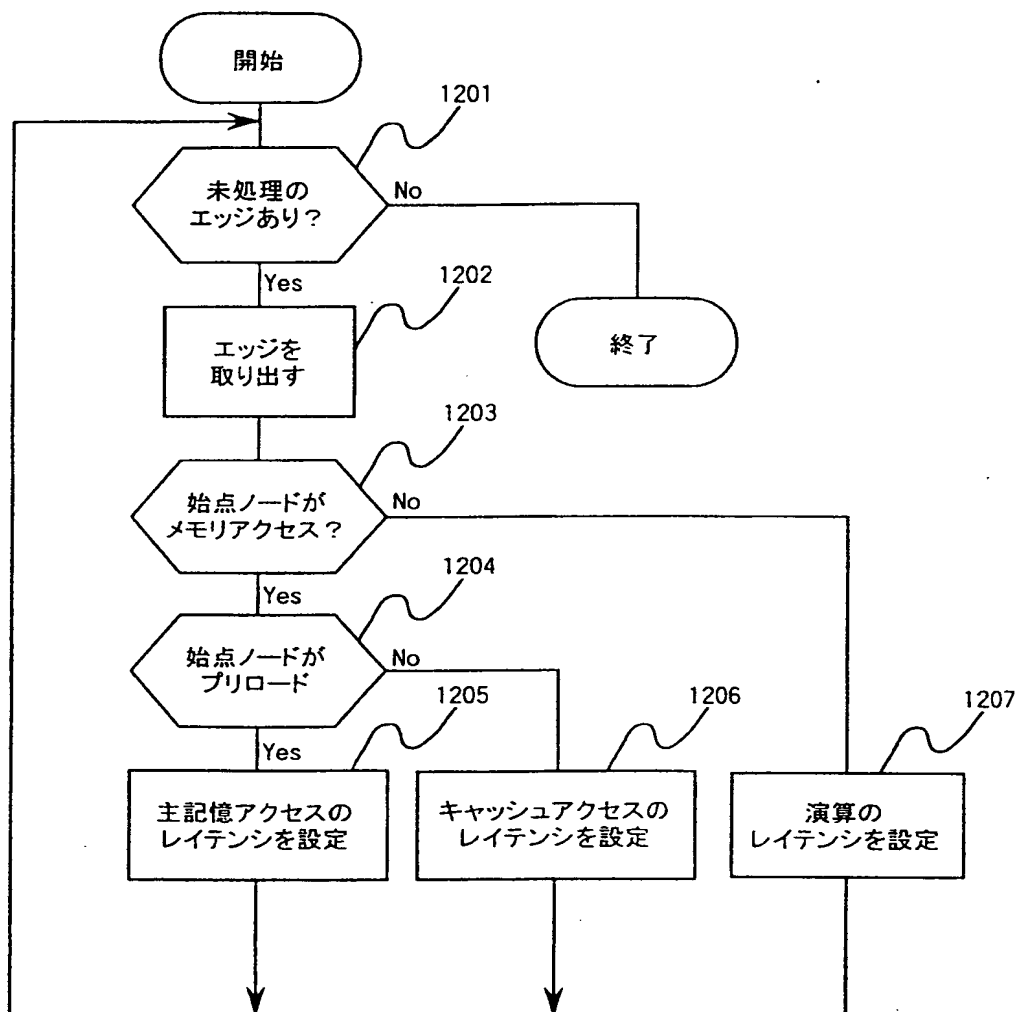


第 11 図



10 / 26

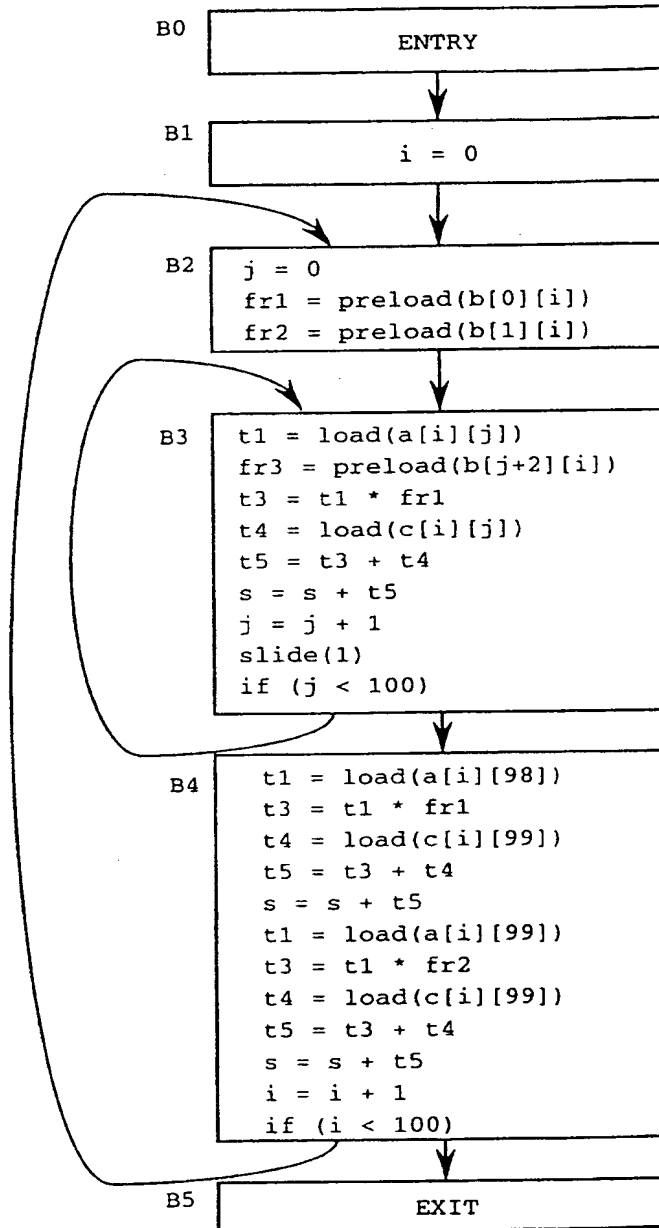
第12図



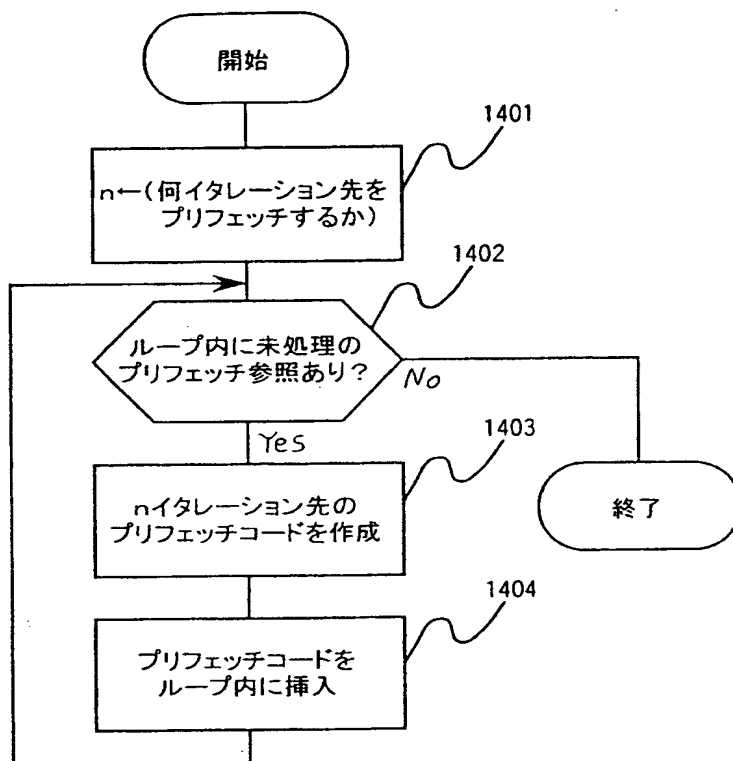


11/26

第13図

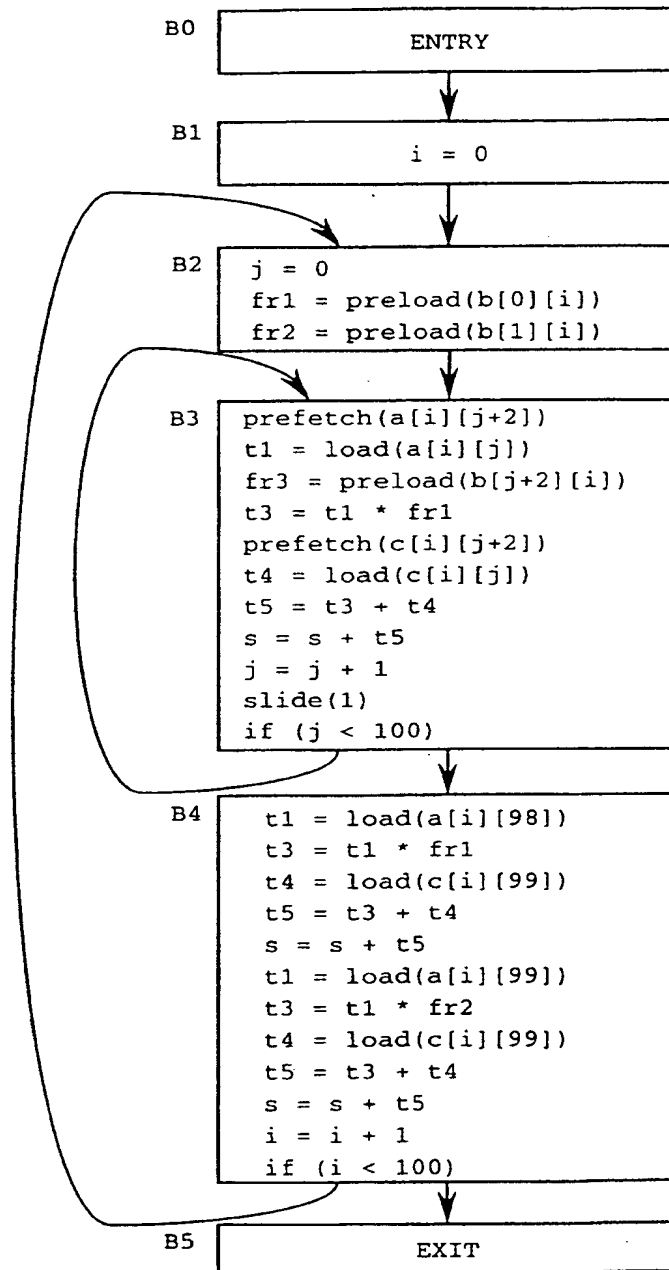


第14図



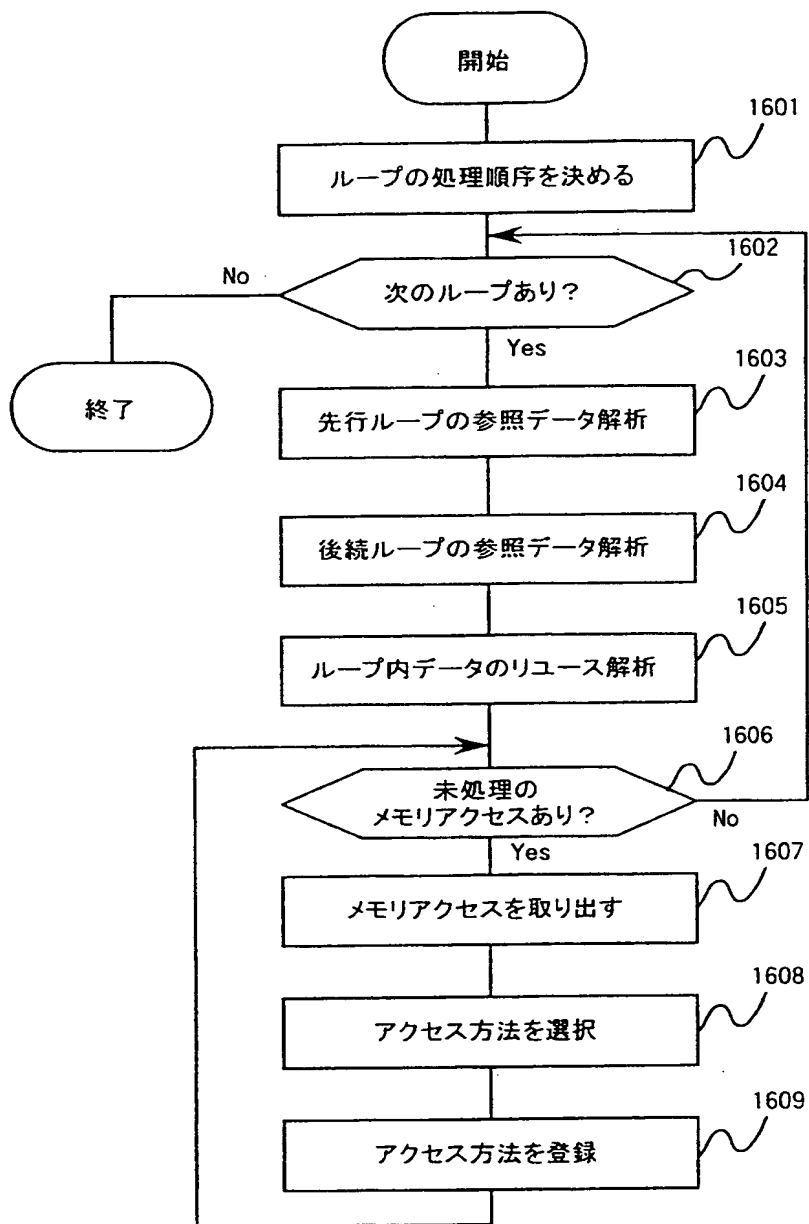
13/26

第15図



14 / 26

第16図



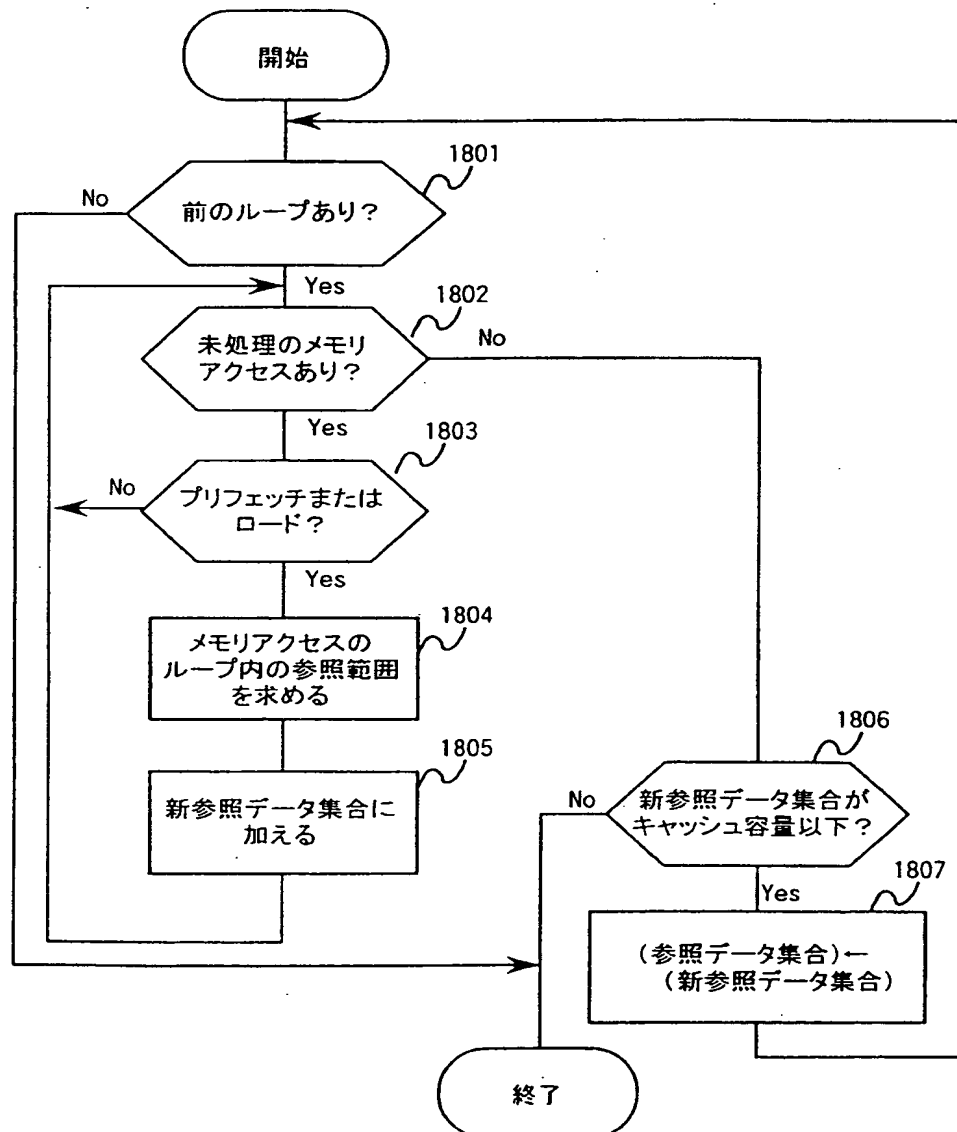
15 / 26

第17図

1701	ループ番号	1	2	3	4
1702	基本ブロック	BB2	BB5	BB8	BB11
1703	前ループ	-	1	2	3
1704	次ループ	2	3	4	-
1705	制御変数	i	i	i	i
1706	制御変数下限値	0	0	0	0
1707	制御変数上限値	99	99	399	399
1708	制御変数増分値	1	1	1	1

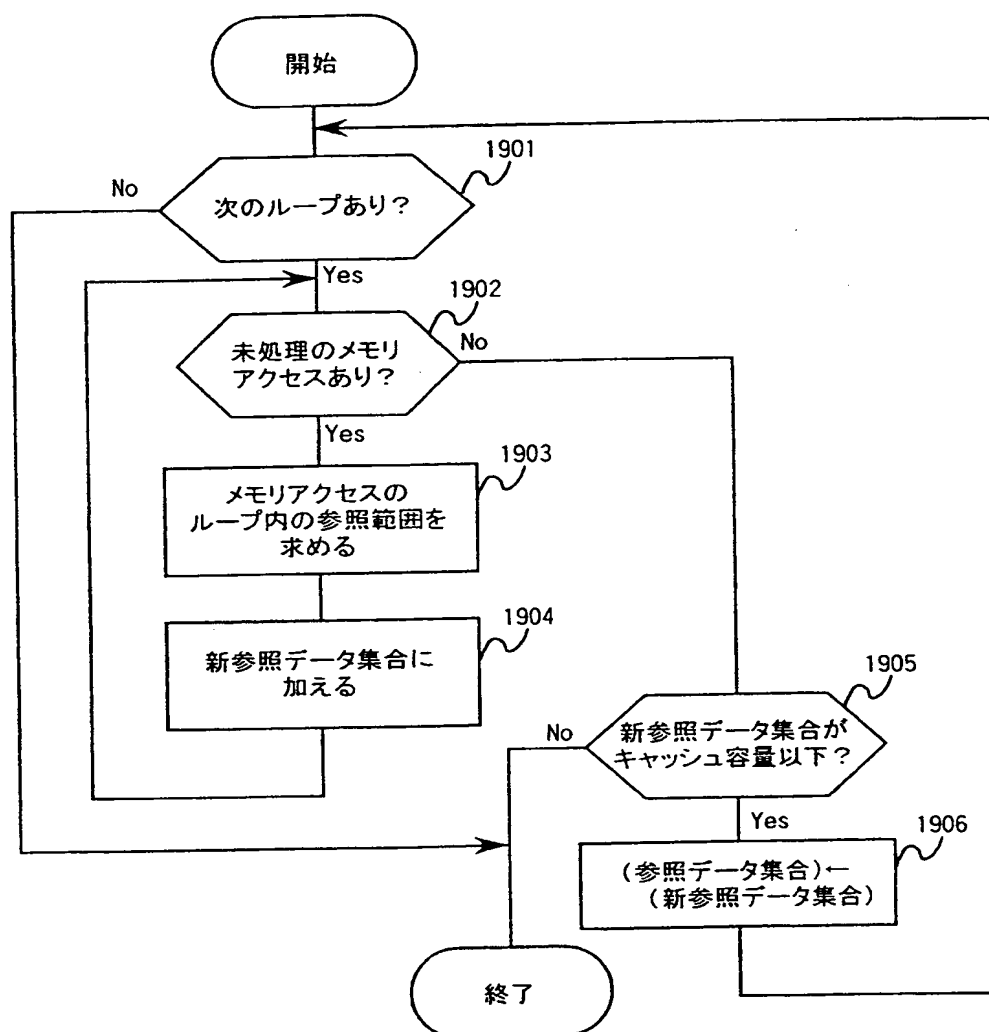
16 / 26

第18図



17 / 26

第19図



18 / 26

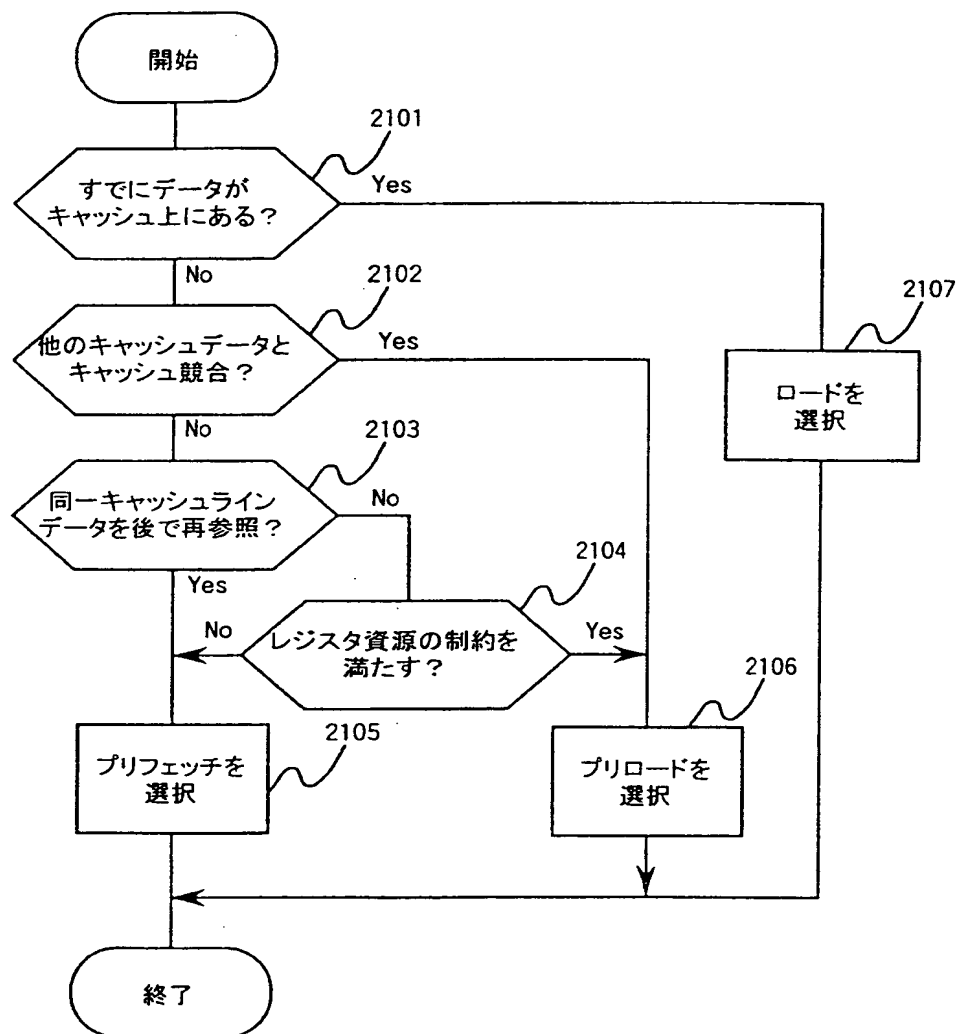
第20図

2001			2002			2003		
メモリアクセス			自己リユース			群リユース		
a[i]			あり			1		
a[i+2]			あり			1		
b[2*i]			あり			なし		
x[i]			あり			なし		
c[x[i]]			なし			なし		
d[i]			あり			2		
d[i+100]			あり			2		
b[2*i+1024]			あり			なし		

2004			2005			2006		
群番号			リーディング参照			他の参照		
1			a[i]			a[i+2]		
2			d[i+100]			d[i]		

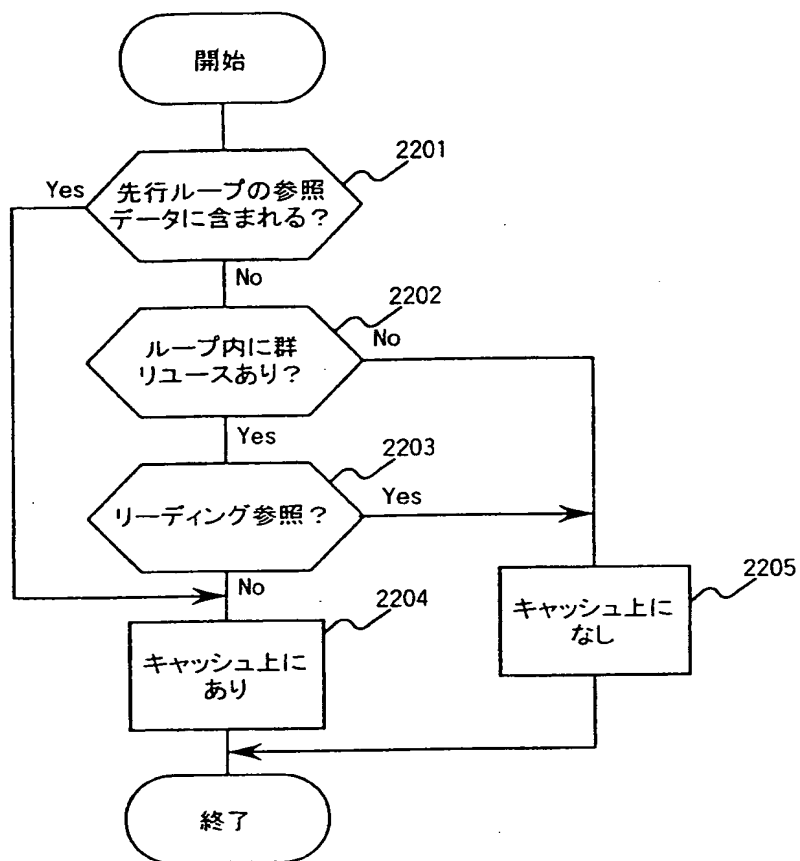


第21図

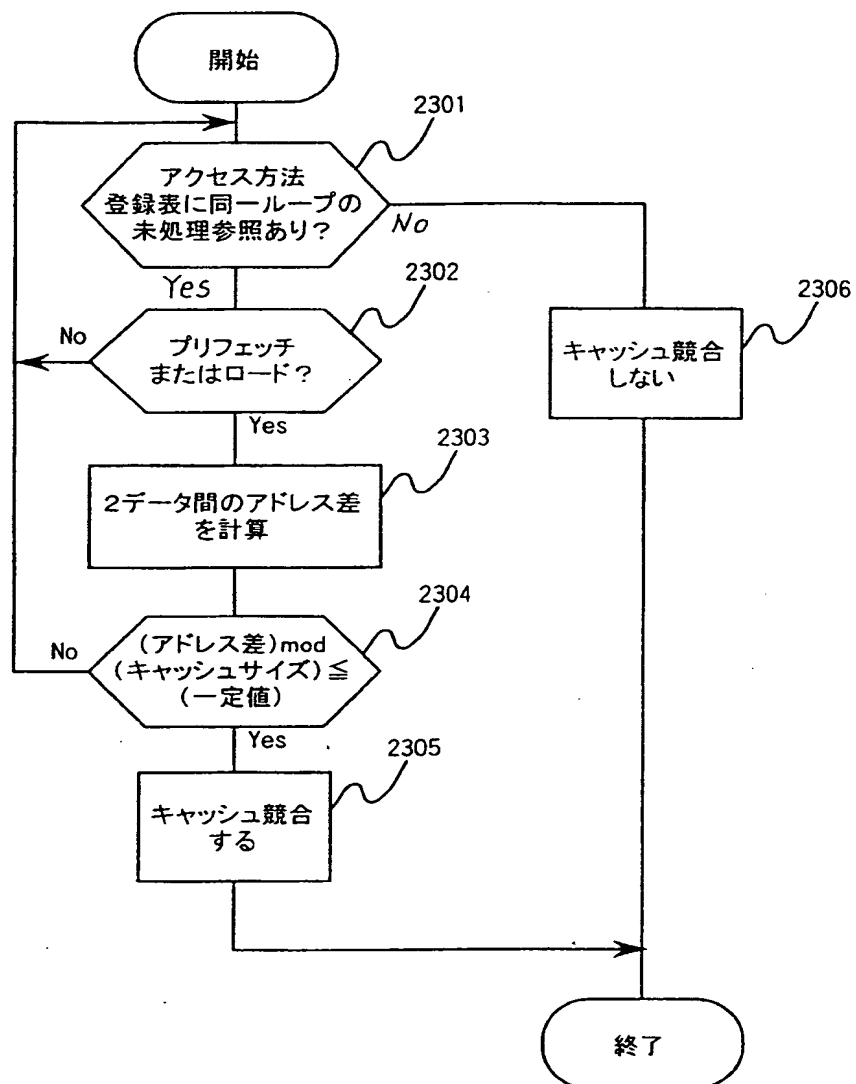


20 / 26

第22図

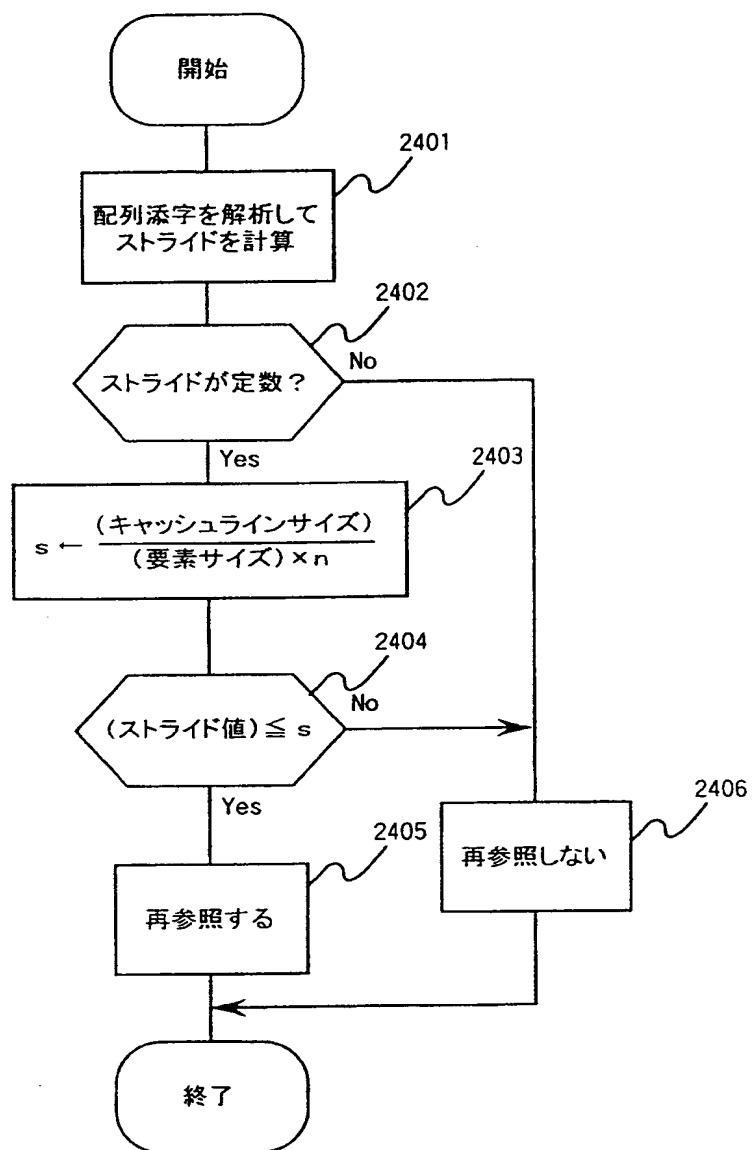


第23図



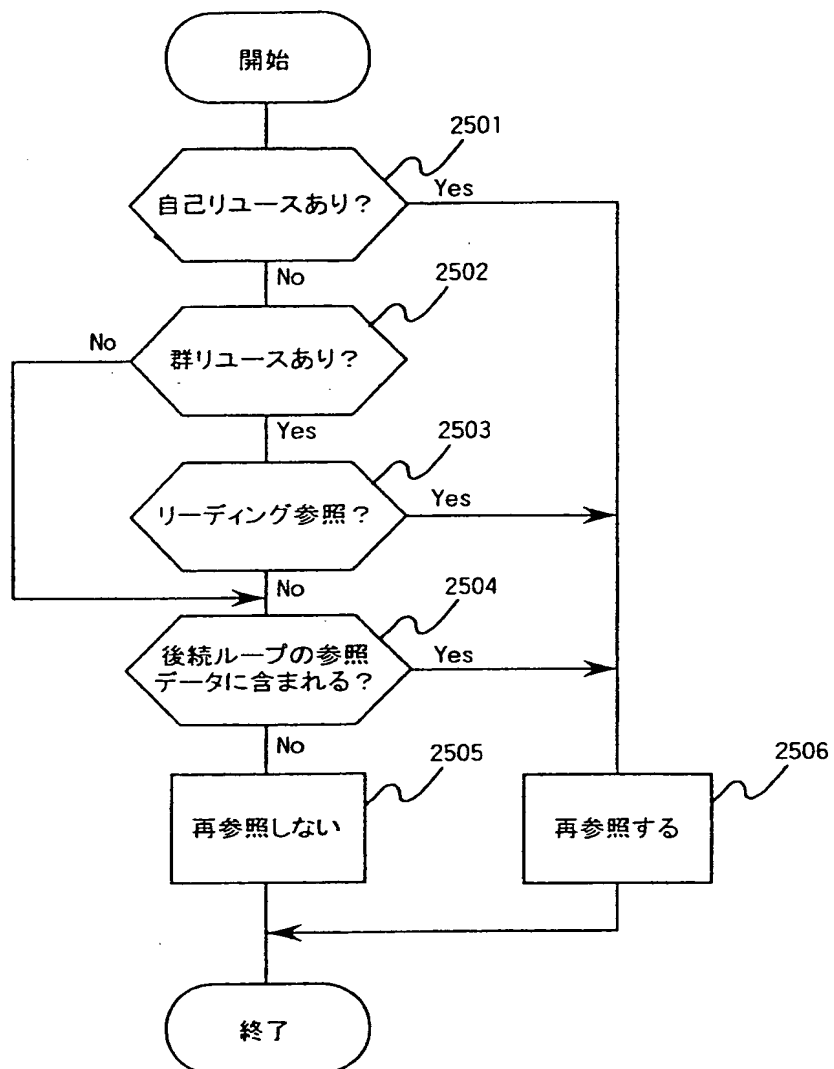
22 / 26

第24図



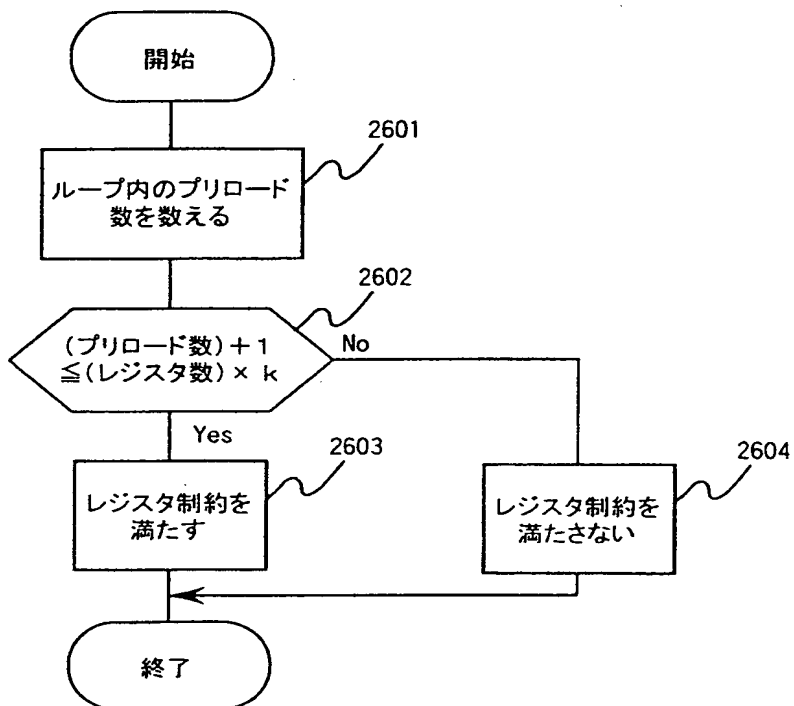
23 / 26

第25図



24 / 26

第26図



25 / 26

## 第27図

```
double  a[200],b[2000],c[100],d[200],s;  
int      x[100];  
void func()  
{  
    int      i;  
  
    for (i = 0; i < 100; i++) {  
        s += a[i] + a[i+2] + b[2*i] + c[x[i]]  
            + d[i] + d[i+100] + b[2*i+1024];  
    }  
}
```

26 / 26

## 第28図

```
double      v1, v2, v3, v4;
double      a[100], b[400], c[400];

void func()
{
    int      i;

    /* LOOP 1 */
    for (i = 0; i < 100; i++) {
        v1 += a[i];
    }

    /* LOOP 2 */
    for (i = 0; i < 100; i++) {
        v2 += a[i] + b[i*4] + c[i*4];
    }

    /* LOOP 3 */
    for (i = 0; i < 400; i++) {
        v3 += b[i];
    }

    /* LOOP 4 */
    for (i = 0; i < 400; i++) {
        v4 += c[i];
    }
}
```



# INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP97/04542

A. CLASSIFICATION OF SUBJECT MATTER  
Int.C1<sup>6</sup> G06F9/45

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int.C1<sup>6</sup> G06F9/45, G06F12/08

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho 1971-1997 Toroku Jitsuyo Shinan Koho 1994-1997  
Kokai Jitsuyo Shinan Koho 1971-1994

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP, 7-295882, A (Hitachi, Ltd.), November 10, 1995 (10. 11. 95) (Family: none)	1-9
A	Papers of Information Proceeding Society of Japan, Vol. 34, No. 5, April 1993, Hiroshi Nakamura et al., "Evaluation of Pseudo-Vector Processor Using Register Window Method (in Japanese)" p.669-680	1-9
A	JP, 4-262458, A (NEC Corp.), September 17, 1992 (17. 09. 92) (Family: none)	1, 2

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

<p>* Special categories of cited documents:</p> <p>"A" document defining the general state of the art which is not considered to be of particular relevance</p> <p>"E" earlier document but published on or after the international filing date</p> <p>"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)</p> <p>"O" document referring to an oral disclosure, use, exhibition or other means</p> <p>"P" document published prior to the international filing date but later than the priority date claimed</p>	<p>"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention</p> <p>"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone</p> <p>"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art</p> <p>"&amp;" document member of the same patent family</p>
--	---

Date of the actual completion of the international search  
March 11, 1998 (11. 03. 98)

Date of mailing of the international search report  
March 24, 1998 (24. 03. 98)

Name and mailing address of the ISA/  
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

## A. 発明の属する分野の分類 (国際特許分類 (IPC))

Int. Cl.<sup>°</sup> G 0 6 F 9 / 4 5

## B. 調査を行った分野

## 調査を行った最小限資料 (国際特許分類 (IPC))

Int. Cl.<sup>°</sup> G 0 6 F 9 / 4 5Int. Cl.<sup>°</sup> G 0 6 F 1 2 / 0 8

## 最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報 1971-1997年

日本国公開実用新案公報 1971-1994年

日本国登録実用新案公報 1994-1997年

## 国際調査で使用した電子データベース (データベースの名称、調査に使用した用語)

## C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
A	J P, 7-295882, A (株式会社日立製作所) 10.11月.1995 (10.11.95) (ファミリーなし)	1-9
A	情報処理学会論文誌, 第34巻, 第5号, 4月.1993, 中村 宏 他「レジスタウィンドウ方式を用いた擬似ベクトルプロ セッサの評価」 p.669-680	1-9
A	J P, 4-262458, A (日本電気株式会社) 17.9月.1992 (17.09.92) (ファミリーなし)	1, 2

☐ C欄の続きにも文献が列举されている。☐ パテントファミリーに関する別紙を参照。

## \* 引用文献のカテゴリー

「A」 特に関連のある文献ではなく、一般的技術水準を示すもの

「E」 先行文献ではあるが、国際出願日以後に公表されたもの

「L」 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献 (理由を付す)

「O」 口頭による開示、使用、展示等に言及する文献

「P」 国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献

「T」 国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの

「X」 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの

「Y」 特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの

「&amp;」 同一パテントファミリー文献

国際調査を完了した日

11.03.98

国際調査報告の発送日

24.03.98

国際調査機関の名称及びあて先

日本国特許庁 (ISA/J P)

郵便番号100-8915

東京都千代田区霞が関三丁目4番3号

特許庁審査官 (権限のある職員)

金田 利規

5B

9292

電話番号 03-3581-1101 内線 3545